UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Values and Ethics in Profession **Subject Code-HU301**
Year: 2nd Year **Semester: Third**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**19L**
	Rapid Technological growth and depletion of resources, Reports of the Club of Rome. Limits of growth: Sustainable development	3
	Energy Crisis: Renewable Energy Resources Environmental degradation and pollution. Eco-friendly Technologies. Environmental Regulations, Environmental Ethics	5
	Appropriate Technology Movement of Schumacher; later developments Technology and developing notions. Problems of Technology transfer, Technology assessment impact analysis.	6
	Human Operator in Engineering projects and industries. Problems of man, machine, interaction, Impact of assembly line and automation. Human centered Technology.	5
2	**Ethics of Profession:**	**9L**
	Engineering profession: Ethical issues in Engineering practice, Conflicts between business demands and professional ideals.	3
	Social and ethical responsibilities of Technologists. Codes of professional ethics. Whistle blowing and beyond.	6
	Profession and Human Values	**8L**
3.		
	Values Crisis in contemporary society Nature of values: Value Spectrum of a good life	3
	Psychological values: Integrated personality; mental health Societal values: The modern search for a good society, justice, democracy, secularism, rule of law, values in Indian Constitution. Aesthetic values: Perception and enjoyment of beauty, simplicity, clarity Moral and ethical values: Nature of moral judgements; canons of ethics; ethics of virtue; ethics of duty; ethics of responsibility.	5

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Mathematics-III **Subject Code-M301**
Year: 2nd Year **Semester: Third**

Module Number	Topics	Number of Lectures
1	**Fourier Series & Fourier Transform**	**8L**
	Introduction	1
	Fourier Series for functions of period 2π, Fourier Series for functions of period 2L	3
	Fourier Integral Theorem, Fourier Transform of a function, Fourier Sine and Cosine Integral Theorem.	1
	Properties of Fourier Transform, Fourier Transform of Derivatives.	1
	Convolution Theorem, Inverse of Fourier Transform.	2
2	**Introduction to Functions of a Complex Variable & Conformal Mapping, Complex Integration, Residue & Counter Integration**	**8L**
	Complex functions, Limit, Continuity and Differentiability, Analytic functions	1
	Cauchy-Riemann Equations, Harmonic function and Conjugate Harmonic function	1
	Construction of Analytic functions: Milne Thomson method.	1
	Simple curve, closed curve, smooth curve & contour, complex Integrals.	1
	Cauchy's theorem, Cauchy-Goursat theorem, Cauchy's integral formula, Cauchy's integral formula	2
3	**Basic Probability Theory, Random Variable & Probability Distributions. Expectation**	**12L**
	Introduction	1
	Conditional probability, Independent events & Multiplication Rule.	1
	Baye's theorem	1
	Random variable	1
	Probability density function & probability mass function.	2
	Expectation & Variance	1
	Binomial & Poisson distributions and related problems.	2
	Uniform, Exponential, Normal distributions and related problems.	3
4	**Partial Differential Equation (PDE) and Series solution of Ordinary Differential Equation (ODE)**	**7L**
	Origin of PDE, its order and degree, concept of solution in PDE.	1
	Different methods: Separation of variables, Laplace & Fourier transform method.	3
	PDE I: One dimensional Wave equation.	1
	PDE II: One dimensional Heat equation	1
	PDE III: Two dimensional Laplace equation	1

Assignment:

Module-1:

1. Write the statement of Fourier integral Theorem.
2. If the Fourier series of function $f(x)$ is given by $a_0 + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$, then a_n is given by?
3. If the Fourier series of function $f(x)$ is given by $a_0 + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$, then b_n is given by?
4. If the Fourier series of function $f(x)$ is given by $a_0 + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$, then a_n is given by?
5. If the Fourier series of function $f(x)$ is given by $a_0 + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$, then b_n is given by?
6. If $F(p)$ is the Fourier transform of $f(x)$, then the Fourier transform of $f(ax)$ is given by?
7. If $F(p)$ is the Fourier transform of $f(x)$, then the Fourier transform of $f(x-a)$ is given by?
8. If $F(p)$ is the Fourier transform of $f(x)$, then the Fourier transform of $f(ax)$ is given by?
9. If $F(p)$ is the Fourier transform of $f(x)$, then the Fourier transform of $f(x-a)$ is given by?
10. Define periodic function
11. Define even function
12. Write the relation between two orthogonal functions.
13. IF convolution of two functions exists then the value of $F\left\{\frac{1}{\sqrt{2®}}\int_{-\infty}^{\infty} f(u)g(x-u)du; p\right\} =$
14. IF convolution of two functions exists then the value of $F\left\{\frac{1}{\sqrt{2®}}\int_{-\infty}^{\infty} f(u)g(x-u)du; p\right\} =$
15. IF convolution of two functions exists then the value of $F\left\{\frac{1}{\sqrt{2®}}\int_{-\infty}^{\infty} f(u)g(x-u)du; p\right\} =$
16. IF convolution of two functions exists then the value of $F\left\{\frac{1}{\sqrt{2®}}\int_{-\infty}^{\infty} f(u)g(x-u)du; p\right\} =$
17. Obtain the Fourier series for the function $f(x) = x^2, -® < x < ®$.
18. Obtain the fourier series for the function $f(x) = \frac{1}{4}(® - x^2), 0 < x < 2®$.
19. Obtain the fourier series for the function $f(x) = \sin ax, -® < x < ®$. a being non-integer value.
20. Obtain the fourier series for the function $f(x) = x, -® < x < ®$.

Module-2:

21. Write Cauchy- Riemann equations for a function $f(z) = u(x, y) + iv(x, y)$.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

22. Write necessary condition for a function $f(z)=u(x,y)+iv(x,y)$ to be analytic.
23. Write necessary and sufficient condition for a function $f(z)=u(x,y)+iv(x,y)$ to be analytic.
24. Write sufficient condition for a function $f(z)=u(x,y)+iv(x,y)$ to be analytic.
25. State Cauchy's integral theorem.
26. Write Cauchy's integral formula.
27. Write type of singularity of the function $\frac{\sin z}{z}$ at $z=0.$
28. Write type of singularity of the function $\frac{z^3}{(z+1)^2(z-5)^4}$ at $z=5.$
29. Write type of singularity of the function $\frac{1}{(z+1)^2(z-3)^2}$ at $z=-1.$
30. Write type of singularity of the function $\frac{z^2}{(z+1)(z-3)^2}$ at $z=3.$
31. Examine that the function $f(x,y)=y^3-3x^2y$ is harmonic or not.
32. Examine that the function $f(x,y)=\frac{1}{2}\log(x^2+y^2)$ is harmonic or not.
33. Examine that the function $f(x,y)=\frac{x-y}{x^2+y^2}$ is harmonic or not.
34. Examine that the function $f(x,y)=2x(1-y)$ is harmonic or not.
35. Evaluate $\int_0^{1+i} z^2dz$, where z is complex number.
36. Evaluate $\int_0^{1+2i}(1+z^2)dz$, where z is complex number.
37. Evaluate $\int_0^{2+i} e^z dz$, where z is complex number.
38. Evaluate $\int_0^{1+i}(z^2+3z+2)dz$, where z is complex number.
39. Find the residue at the poles of $f(z)=\frac{\cot \circledR z}{(z-a)^2}.$
40. Find the residue at the poles of $f(z)=\frac{z^2-2z}{(z+1)^2(z^2+4)}.$
41. Find the residue of $f(z)=\frac{z^3}{z^2-1}$ at $z=\infty.$
42. Find the residue of $f(z)=\frac{e^z}{z\sin mz}$ at $z=0.$

Module-3:

1. If for two events A and B we have the following probabilities:

$P(A)=P(A|B)=\frac{1}{4}; P(B|A)=\frac{1}{2}$. Then check A and B are independent or not.

2. If $P(A\cap B)=\frac{1}{2}, P(\overline{A}\cap\overline{B})=\frac{1}{2}$ and $2P(A)=P(B)=p$, then find the value of p.
3. If for two events A and B we have the following probabilities:
   $P(A)=P(A|B)=\frac{1}{4}; P(B|A)=\frac{1}{2}$. Then find $P(\overline{A}|B)=$.
4. If $P(A\cap B)=\frac{1}{2}, P(\overline{A}\cap\overline{B})=\frac{1}{3}$ and $P(A)=P(B)=p$, then find the value of p.
5. If A and B are any two events and $P(A)=p_1; P(B)=p_2; P(A\cap B)=p_3$. Then $P(\overline{A}\cup\overline{B})=$
6. If A and B are any two events and $P(A)=p_1; P(B)=p_2; P(A\cap B)=p_3$. Then $P(\overline{A\cup B})=$
7. If A and B are any two events and $P(A)=p_1; P(B)=p_2; P(A\cap B)=p_3$. Then $P(\overline{A}\cap\overline{B})=$
8. If A and B are any two events and $P(A)=p_1; P(B)=p_2; P(A\cap B)=p_3$. Then $P(\overline{A}\cup B)=$
9. State Baye's theorem for mutually disjoint events.
10. If $f(x)=\begin{cases} ke^{-2x} & x\geq 0 \\ 0 & otherwise \end{cases}$, then what will be the value of k for which $f(x)$ be probability density function?
11. If $f(x)=\begin{cases} x & 0<x<1 \\ 0 & otherwise \end{cases}$, then $f(x)$ is probability density function or not?
12. If $f(x)=\begin{cases} ke^{-x} & x\geq 0 \\ 0 & otherwise \end{cases}$, then what will be the value of k for which $f(x)$ be probability density function?
13. If $f(x)=\begin{cases} k\left(1-e^{-x}\right)^2 & x\geq 0 \\ 0 & otherwise \end{cases}$, then what will be the value of k for which $f(x)$ be probability density function?
14. Write the formula for mathematical expectation of a discrete random variable X with probability mass function $f(x)$.
15. Write the formula for mathematical expectation of a continuous random variable X with probability density function $f(x)$.
16. Write the formula for mathematical expectation of a discrete random variable X with probability mass function $f(x)$.
17. Write the formula for mathematical expectation of a continuous random variable X with probability density function $f(x)$.

18. A card is drawn from pack of 52 cards, find the probability of getting a king or a heart or a red card?
19. A card is drawn from a pack of 52 cards, if the value of faces cards 10, aces cards 1 and other according to denomination, find the expected value of the no. of point on the card.
20. A bag contains 10 red and 15 white balls. Two balls are drawn in succession. What is the probability that one of them is white and other red?
21. State Bayes' theorem.
22. A and B take turns in throwing two dice on the understanding that the first to throw 9 will be awarded a prize. If A has the first turn, show their respective chances of winning are in the ratio 9 : 8.
23. Three groups of children contain respectively 3 girls and 1boy; 2 girls and 2 boys; 1 girls and 3 boys, One child is selected at random from each group. Find the chance of selecting 1 girl and 2 boys.
24. A manufacturer supplies quarter horsepower motors in lots of 25. A buyer, before taking a lot, tests at random a sample of 5 motors and accepts the lot if they are all good; otherwise he rejects the lot. Find the probability that :(i) he will accept a lot containing 5defective motors ;(ii) he will reject a lot containing only one defective motors.
25. In an examination with multiple-choice questions, each question has four, out of which one is correct. A candidate ticked the answer either by his skill or by copying from his neighbours, The probability of guess is 1/3, copying is 1/6. The probability of correct answer by copying is 1/8. If a candidate answers a question correctly find the probability that he know the answer.
26. An urn contains 10 white and 3 black balls. Another urn contains 3 white and 5 black balls. Two balls are drawn at random from first urn and placed in the second urn and then one ball is taken at random from the latter. What is the probability that it is a white ball ?
27. Define the random variable, Explain the types of random variable with example.
28. A can hit a target 4 times in 7 shots, B 3 times in 5 shots and C three times in 5 shots. All of them fire one shot each simultaneously at the target. What is the probability that (i) 2 shots hit (ii) At least two shots hit ?
29. The probability that a student A solves a mathematics problem is 2/5 and the probability that a student B solves the problem is 2/3. What is the probability that (a) the problem is not solved (b) the problem is solved (c) both A and B solve the problem.
30. A company has four production section S_1, S_2, S_3& S_4 which contribute 30%, 20% 22% & 28% respectively produced 1%, 2%, 3% & 4% defective units, if a small unit is selected random & found to be defective, what is the probability that the unit selcected has came from (a) Section S_1 (b) Section S_4
31. From a city population, the probability of selecting a male or a smoker is 7/10, a male smoker is 2/5 and a male if a smoker is already selected is 2/3, find the probability of selecting (a) non-smoker (b) a male (c) a smoker if a male is first selected.
32. There are two bags A and B. A contains n white and 2 black balls & B contains 2 white and n black balls, one of the two bags is selected at random and two balls are drawn from it without replacement. If the both balls are drawn are white and the probability that the bag A was used to drawn the ball is 6/7. Find the value of n.

Module-4:

1. Bessel function of order $p = \pm\frac{1}{2}$, show that $J_{1/2}(x) = \sqrt{2/®x}\sin x$ and $J_{-1/2}(x) = \sqrt{2/®x}\cos x$.
2. Determine the order p of the following Bessel equation:
   a) $x^2y'' + xy' + (x^2 - 9)y = 0$

b) $x^2 y'' + xy' + x^2 y = 0$

3. Solve the following heat flow problem:

$$\frac{\partial u}{\partial t} = 7\frac{\partial^2 u}{\partial x^2} \quad , \quad 0 < x < ® \quad , \quad t > 0.$$

$$u(0,t) = u(®\ ,t) = 0\ ,\ \ t > 0\ ,$$

$$u(x,0) = 3\sin 2x - 6\sin 5x \ \ , \ 0 < x < \pi.$$

4. Prove that F satisfies the Laplace's equation: $F = Cz^n$

$$\nabla^2 F = \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} = 0$$

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Advanced OOPs using C++ Subject Code-CS301

Year:2nd Year Semester: Third

Module Number	Topics	Number of Lectures
1	**Introduction**	**5L**
	1. Basics of OOP, Features; Structure of C++ program; Class and object; Concept of Constructor& destructor; Abstraction 2. Encapsulation; Inheritance; 3. Static and dynamic binding;Polymorphism.	2L 1L 2L
2	**Exception Handling**	**5L**
	1. Exception handling mechanism; throwing, catching, rethrowing mechanism; 2. Multiple catch statement; Nested try-catch block; 3. Exception in constructor & destructor; exceptions in operator overloaded functions.	2L 2L 1L
3	**Template**	**6L**
	1. Class template; Member function inclusion; Class template with different parameter; 2. Function template; Function template with multiple parameters; 3. Overloading of template function; member function template.	3L 2L 1L
4	**Console I/O operations**	**6L**
	1. C++ streams; C++ stream classes; 2. Unformatted I/O operations; 3. Formatted I/O operations; 4. Managing output with Manipulators.	1L 2L 2L 1L
5	**Working with Files**	**10L**
	1. Data File Handling: Need for a data file, Types of data files – Text file and Binary file; 2. Text File: Basic file operations on text file: Creating/Writing text into file, reading and manipulation of text from an already existing text File (accessing sequentially). 3. Binary File: Creation of file, Writing data into file, Searching for required data from file, Appending data to a file, Insertion of data in sorted file, Deletion of data from file, Modification of data in a file; opening and closing files; classes	2L 4L 4L

	for file stream operations; Error handling during file operations; command line arguments.	
6	**Standard Template Library**	**4L**
	1. Components of STL; Containers, Iterator; 2. Applications of container classes.	2L 2L
	Standard Functions Library	**4L**
	1. C-based I/O functions (fflush, fgetc, ferror, fscanf, fprintf etc.); Time, Date, Localization functions (asctime, clock, ctime, difftime, localtimemktime,strftime etc.); 2. Dynamic memory allocation functions (calloc, malloc, realloc, free).	2L 2L
7	**String Manipulation**	**4L**
	1. The String class; Creating String object; Manipulating strings; 2. Relational operations on strings; String comparison characteristics, swapping; Accessing characters in strings.	1L 3L
Total Number Of Lectures = 44		

Faculty In-Charge HOD, CSE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Advanced OOPs using C++ Subject Code-CS301
Year:2nd Year Semester: Third
Assignment:

Module-1: Introduction

1. Write a program in which a class has three data members: name, roll no, marks of 5 subjects and a member function Assign() to assign the streams on the basis of table given below:

Avg. Marks	Stream
90% or more	Computers
80% - 89%	Electronics
75% - 79%	Mechanical
70% - 74%	Electrical

2. What is virtual base class? Write a small code to explain it.Why pure virtual functions are needed?Define a function area() to compute the area of objects of different classes – triangle, rectangle, square. Invoke these in the main program. Comment on the binding (static or dynamic) that takes place in your program.

Module-2: Exception Handling

1. Explain exceptions with different types. How exception is handled in C++?Write down the steps by which we can handle the exceptions with a proper example.
2. How we can restrict a function to throw only certain specified exception in C++. Explain with proper example.When we use catch(…) handler? Explain with a proper example.Write programs that demonstrate how certain exception types are not allowed to be thrown.

Module-3: Template

1. What do you mean by generic programming?Explain class template and function template with proper example.A template can be considered as a kind of macro. Then what is the difference between them?
2. What is the difference between class template and template class?Write a function template to perform linear search/ binary search/ bubble sort/ merge sort/ selection sort in an array.Write a C++ program where template function is overloaded.Explain inline function template.

Module-4: Console I/O operations

1. Differentiate get(), getline() and write() function with proper example.What do you mean by stream? Write down its different types of stream with proper example.What is streambuf? Explain stream.Why it is necessary to include the file iostream in all C++ program?
2. A. What the following statement do?
   i. cout.write(s1,m).write(s2,n)
   ii. cout.write(line, size)
   iii. cout.precision(a)

   B. How do the following two statements differ in operation?
   a. Cin>>a;
   b. Cin.get(a);

Module-5: Working with Files

1. Write a C++ program to write number 1 to 100 in a data file NOTES.TXT.Explain how while(f1) statement detects the end of file that is connected to f1 stream.
2. Explain the difference between normal text file and binary file. What are the advantages of saving data in binary form?How many file objects are required to do the following operation?
   iv. To process four files sequentially.
   v. To merge two files into third file.

Module-6: STL

1. What are the different types of algorithm in STL? Explain.What are the applications of container class?How list works in STL? Explain with proper example.What do you mean by function objects? How you use function objects in algorithm?
2. A table gives a list of car models and the number of units sold in each type in a specified period. Write a program to store this table in a suitable container, and to display interactively the total value of a particular model sold, given the unit-cost of that model.

Module-7: String Manipulation

1. What is the importance of using string class? What do you mean by C-Style string?Write a C++ program to create string objects in C++.
2. Write a program that reads the name "RAMAN KUMAR BANERJEE" from keyboard into three separate string objects and then concatenates them into a new string object using
    i. + operator
    ii. append() function

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Analog & Digital Electronics **Subject Code-CS302**
Year: 2nd Year **Semester: Third**

Module Number	Topics	Number of Lectures
1	**Analog Electronics:**	**9L**
	1. Different Classes of Amplifiers - (Class-A, B, AB and C - basic concepts, power, efficiency. .	2
	2. Recapitulation of basic concepts of Feedback and Oscillation	1
	3. Phase Shift, Wein Bridge oscillators	2
	4. Astable & Mono stable Multi vibrators ; Schimtt Trigger circuits ; 555 Timer.	4
2	**Number System, Combinational Circuit:**	**11L**
	1. Binary Number System & Boolean Algebra (recapitulation)	1
	2. BCD, ASCII, EBDIC, Gray codes and their conversions;	1
	3. Signed binary number representation with 1's and 2's complement methods.	1
	4. Binary arithmetic, Venn diagram, Boolean algebra (recapitulation).	1
	5. Representation in SOP and POS forms.	1
	6. Minimization of logic expressions by algebraic method.	2
	7. Combinational circuits: Adder and Subtractor circuits (half & full adder & subtractor)	2
	8. Encoder, Decoder, Comparator, Multiplexer, De-Multiplexer and Parity Generator	2
3.	**Sequential Circuits:**	**10L**
	1. Sequential Circuits: Basic Flip-flop & Latch	1
	2. Flip-flops -SR, JK, D, T and JK Master-slave Flip Flops	2
	3. Registers (SISO,SIPO,PIPO,PISO)	2
	4. Ring counter, Johnson counter	1
	5. Basic concept of Synchronous and Asynchronous counters (detail design of circuits excluded)	2
	6. Design of Mod N Counter.	2
4	**Conversion & Logic Families:**	**6L**
	1. A/D and D/A conversion techniques – Basic concepts (D/A): R-2-R only	2

	2. A/D: successive approximation	2
	3. Logic families- TTL, ECL, MOS and CMOS - basic concepts.	2
Total Number Of Hours = 36		

Faculty In-Charge HOD, CSE Dept.

Assignment:

Module-1:

1. Explain what is 555 timer?
2. Explain why normally control terminal of 555 timer is connected to ground through a 0.01µF bypass capacitor?
3. Why mono stable mode is called "one-shot" pulse generator?
4. What are barkhausen criteria?
5. What is loop gain? Derive the gain for sustained oscillation.

Module-2:

1. Add the following decimal numbers using the 2's complement method:
   i) -46 and +22 ii) -55 and -16 iii) +23 and -22
2. Add the following BCD numbers:
   a) 10011100 and 00110100 b) 01011001 and 01010010
3. Subtract the following using the 9's complement method:
   a) 786-427 b) 473-438 c) 357-294

4.a) Convert the following binary number to gray code:
   i) 10110 ii) 011011 iii) 1100110
   b) Convert the following gray code to binary number:
   i) 011011 ii) 11011 iii) 011110

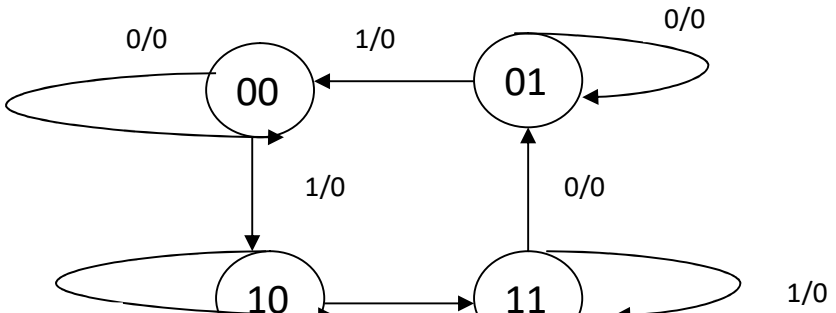
5. Simplify the following Boolean expression:
   a) $Y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\ \ \bar{C}$
   b) $Y = (\bar{A} + B)\overline{(A + B)}$
   c) Y= A[B+C($\overline{AB+AC}$)]
6. Simplify the expression $Y = \sum_m(0,2,3,5,7,8,10,11,14,15)$ using K-map method.
7. Simplify the expression $Y = \prod(1,5,6,7,11,12,13,15)$ using the K-map method.
8. Obtain the minimal sum of products expression for the following function:
   $f(A,B,C,D) = \sum(3,7,11,12,15) + \sum_d(1,4,5,10,12,14)$

9. Design a half adder circuit using NAND gates.
10. Excess 3 code is self complementing code – Explain.
11. Design a full adder circuit using basic gates and half adder.
12. Implement a 16-to-1 multiplexer using two 8-to-1 multiplexer.
13. Implement the following function using a 8-to-1 and 1 4 to 1 multiplexer:
    $F(A,B,C,D) = \sum(0,1,3,4,8,9,15)$
14. Design a full subtractor circuit using a multiplexer.
15. Design a 2 bit comparator with basic gates.
16. Write down the differences between combinational logic circuit and sequential logic circuit.
17. Design an odd parity generator circuit.

18. Write a short note on octal to binary encoder.
19. Design Full Adder by decoder.
20. Difference between decoder and de multiplexer.
21. Define weighted code and non weighted code.

Module-3:

1. What is meant by edge triggering? Give the difference between positive and negative edge triggering.
2. Explain the operation of master slave JK flip flop and show how the race around condition is eliminated in it.
3. Explain the function of a D flip flop using a suitable diagram and discuss how it works as a latch.
4. Perform the conversion from T flip-flop to SR flip-flop.
5. Using proper truth table and logic diagram find the characteristic equation and excitation table of SR flip flop.
6. Using proper truth table and logic diagram find the characteristic equation and excitation table of JK flip flop.
7. Using proper truth table and logic diagram find the characteristic equation and excitation table of D flip flop.
8. Using proper truth table and logic diagram find the characteristic equation and excitation table of T flip flop.
9. Write the difference between latch and flipflop.
10. Write the difference between asynchronous counter and synchronous counter.
11. Explain the operation of a 4-bit asynchronous ripple counter(up). Show the output waveform also.
12. Explain the operation of a 4bit synchronous counter(up) using proper logic diagram and truth table.
13. What is register? What is shift register? What are the types of shift register? Sketch the block diagrams of each type.
14. Explain the operation of shift left serial input serial output register with suitable logic diagram.
15. Explain the operation of shift right serial input serial output register with suitable logic diagram.
16. Explain the operation of parallel input serial output register with suitable logic diagram and waveforms.
17. Explain the operation of ring counter with suitable logic diagram and waveform.
18. Explain the operation of Johnson counter with suitable logic diagram and waveform.
19. Design a MOD-6asynchronous counter.
20. Design a MOD-12synchronous counter.
21. Design a synchronous 3 bit counter using all negative edge triggered JK type flipflop which will count following sequence-0,2,4,6,0 when the control input X is 0 and count 1,3,5,7,1 when the control input X is 1.
22. Design a sequential circuit that implement the following state diagram.



23. Explain the operation of a 4-bit asynchronous ripple counter(up and down both).
24. Explain the operation of a 4-bit bit directional shift register.

Module-4:

1. Explain R 2R Ladder type converter.
2. Explain Flash type converter.
3. Define CMOS Logic family.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Data Structure & Algorithm **Subject Code-CS303**
Year: 2nd Year **Semester: Third**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**5L**
	1. Why we need data structure? Concepts of data structures: a) Data and data structure b) Abstract Data Type and Data Type. Algorithms and programs, basic idea of pseudo-code. .	1
	2. Algorithm efficiency and analysis, time and space analysis of algorithms – order notations.	4
	Linear data structure:	
2	**Array:**	**2L**
	1. Different representations – row major, column major. Sparse matrix - its application and usage. Array representation of polynomials.	2
3	**Linked List:**	**7L**
	1. Singly linked list, circular linked list, doubly linked list, linked list representation of polynomial and applications.	7
4	**Stack and Queue:**	**6L**
	1. Stack and its implementations (using array, using linked list), applications.	2
	2. Queue, circular queue, dequeue. Implementation of queue- both linear and circular (using array, using linked list), applications.	4
5	**Recursion:**	**3L**
	1. Principles of recursion – use of stack, differences between recursion and iteration, tail recursion.	1
	2. Applications - The Tower of Hanoi, Eight Queens Puzzle.	2
	Non Linear data structure:	
6	**Trees:**	**8L**
	1. Basic terminologies, forest, tree representation (using array, using linked list). Binary trees - binary tree traversal (pre-, in-, post- order), threaded binary tree (left, right, full) - non-recursive traversal algorithms using threaded binary tree, expression tree.	4
	2. Binary search tree- operations (creation, insertion, deletion, searching). Height balanced binary tree – AVL tree (insertion, deletion with examples only). B- Trees – operations (insertion, deletion with examples only	4
7	**Graphs:**	**5L**
	1. Graph definitions and concepts (directed/undirected graph, weighted/un-weighted edges, sub-graph, degree, cut-vertex/articulation point, pendant node, clique, complete graph, connected components – strongly connected component, weakly connected component, path, shortest path, isomorphism). Graph representations/storage implementations – adjacency matrix, adjacency list, adjacency multi-list.	1

	2. Graph traversal and connectivity – Depth-first search (DFS), Breadth-first search (BFS) – concepts of edges used in DFS and BFS (tree-edge, back-edge, cross-edge, forward-edge), applications.	2
	3. Minimal spanning tree – Prim's, Kruskal and Dijkstraalgorithm (basic idea of greedy methods).	2
8	**Sorting, Searching and Hashing Technique:**	
	Sorting Algorithms:	6L
	Bubble sort and its optimizations, insertion sort, shell sort, selection sort, merge sort, quick sort, heap sort (concept of max heap, application – priority queue), radix sort.	6
	Searching:	2L
	Sequential search, binary search, interpolation search.	2
	Hashing:	2L
	Hashing functions, collision resolution techniques.	2
Total Number Of Hours = 46		

Faculty In-Charge HOD, CSE Dept.

Assignment:

Module-1(Introduction):

1. DefineAbstractData Type, big oh, big omega, theta notationoftimecomplexity.
2. Findthetotalfrequency count of following code.

```
for send=1ton do
        forreceive=1tosenddo
                forack=2toreceivedo
                        message=send-(receive+ack)
                        ack=ack-1
                        send=send+1
                end
        end
end
```

Module-2 (Linear data Structure):

1. Write a function to insert a element after 4^{th} position in an array.
2. Write a function to insert a element before 4^{th} position in a single linked list
3. Write a function to insert a element after a particular data element 4 in a doubly linked list.
4. Write a function to concatenate two circular linked list.
5. Write a function to implement stack and queue using linked list.
6. Convert infix to prefix and postfix.
   A+B+C-D/E*R(S*T)/W+G
7. Define tail and tree recursion, explain them with example.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Module-3(Non-linear data structure):

1. Why AVL tree is required?
2. Construct the AVL tree.
   B,D,A,G,H,R,J,T,C,Y,X
3. Write a short note on B-Tree.
4. Write an algorithm of DFS and Dijkstra algorithm.

Module-4(Sorting, Searching and Hashing):

1. Explain quick and radix sort with example.
2. Why binary search is better than linear search.
3. Write down different techniques of collision resolution techniques.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Computer Organization** Subject Code: **CS304**

Year: **2nd Year** Semester: **Third**

Module Number	Topics	Number of Lectures
1	**Introduction to digital computer organization**	**7L**
	1. Concept of basic components of a digital computer, High level view of computer. 2. Commonly used number systems 3. Fixed and floating point representation of a number 4. Booth's Algorithm – restoring and non restoring.	6L
	5. What are the H/W resources that we will need in computer 6. Conversion of high level code to m/c level language.	1L
2	**CPU design**	**3L**
	1. Basic organization of the stored program computer and operation sequence for execution of a program. 2. Description of ALU, Design of circuit of a small scale CPU	2L
	3. Fetch, decode and execute cycle, Concept of operator, operand, registers and storage, Instruction format. Instruction sets and addressing modes.	1L
3	**CPU design - Timing and control**	**2L**
	1. Timing diagram and control design	2L
4	**Micro programmed control**	**5L**
	1. Concepts of Micro operations 2. Horizontal and vertical micro-program 3. Optimization of hardware resources for designing of micro-programmed control unit	5L
5	**Pipeline concept**	**3L**
	1. Instruction and Arithmetic Pipelining, 2. Synchronous and Asynchronous pipeline 3. Solving problems on pipeline speed-up, efficiency, throughput	2L
6	**Memory Organization**	**5L**
	1. Static and dynamic memory, Memory hierarchy, Associative memory, Bare machine	3L
	2. Memory unit design with special emphasis on implementation of CPU-memory interfacing.	2L

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Computer Organization** Subject Code: **CS304**
Year: **2nd Year** Semester: **Third**

7	**Cache memory Architecture**	**4L**
	1. Cache memory organizations, Set associative cache	1L
	2. Techniques for reducing cache misses	1L
	3. Discussion on Buffer cache	2L
8	**RAM architecture**	**2L**
	1. Basic concepts of architecture of RAM	2L
9	**Discussion on DRAM & SRAM**	**3L**
	1. Architecture of Static Ram and Dynamic Ram	2L
	2. Difference between SRAM and DRAM	1L
10	**I-O subsystem organization**	**3L**
	1. Concept of handshaking, Polled I/O	1L
	2. Interrupt and DMA	2L
Total Number Of Hours = 37L		

Faculty In-Charge HOD, CSE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Computer Organization** Subject Code: **CS304**
Year: **2nd Year** Semester: **Third**

Assignments:-

Unit 1:-

1. What is the role of operating system?
2. Discuss the basic organization of digital computer.
3. Convert (1B3F)16 → (?)8
4. Why do we need 2's compliment method?
5. Briefly explain IEEE 754 standard format for floating point representation in single precision.
6. Using Booth's algorithm multiply (– 12) and (+ 6).
7. Explain various assembler directives used in assembly language program.
8. Write +710 in IEEE 754 floating point representation in double precision.

Unit 2 + Unit 3 + Unit 4:-

1. What are the advantages of micro programming control over hardwired control?
2. Write down the control sequence for Move (R1), R2.
3. Define hardwired control.
4. Discuss the principle of operation of a micro programmed control.
5. Write the control sequence for execution of the instruction Add(R3), R1.
6. Give the organization of typical hardwired control unit and explain the functions performed by the various blocks.
7. Explain the multiple bus organization in detail.
8. What is microprogrammed sequencer?
9. Design a basic ALU which can perform 8 different arithmetic and logical operations.
10. Design the basic block diagram of Intel microprocessor 8085 and discuss the working principle of the different parts of it.

Unit 5 :-

1. What is pipelining?
2. What are the major characteristics of a pipeline?
3. What is a pipeline hazard?
4. What is the use of pipelining?
5. What are the remedies commonly adopted to overcome/minimize these hazards.
6. What is the ideal speedup expected in a pipelined architecture with n stages. Justify your answer.
7. Draw the structure of two stage instruction pipeline.

Unit 6 + Unit 7 + Unit 8 + Unit 9 :-

1. Define Memory Access time for a computer system with two levels of caches.
2. How to construct an 8M * 32 memory using 512 K * 8 memory chips.
3. Write two advantages of MOS device.

4. List the factors that determine the storage device performance.
5. What will be the width of address and data buses for a 512K * 8 memory chip?
6. Define memory cycle time.
7. What is RAM?
8. What is cache memory?
9. Explain virtual memory.
10. List the various semiconductors RAMs?
11. Define DRAM's.
12. What is ROM?
13. Give the format for main memory address using direct mapping function for 4096 blocks in main memory and 128 blocks in cache with 16 blocks per cache.
14. Give the format for main memory address using associative mapping function for 4096 blocks in main memory and 128 blocks in cache with 16 blocks per cache.
15. Give the format for main memory address using set associative mapping function for 4096 blocks in main memory and 128 blocks in cache with 16 blocks per cache.
16. Define Hit and Miss rate?
17. What are the enhancements used in the memory management?
18. Define latency time.
19. A computer system has a main memory consisting of 16 M words. It also has a 32Kword cache organized in the block-set-associative manner, with 4 blocks per set and 128 words per block.
20. How will the main memory address look like for a fully associative mapped cache?
21. A digital computer has a memory unit of 64K*16 and a cache memory of 1K words. The cache uses direct mapping with a block size of four words. How many bits are there in the tag, index, block and word fields of the address format? How many blocks can the caches accommodate?
22. Define the terms "spatial locality" and "temporal locality", and explain how caches are used to exploit them for a performance benefit. Be specific in the different ways that caches exploit these two phenomena.
23. Suppose physical addresses are 32 bits wide. Suppose there is a cache containing 256K words of data (not including tag bits), and each cache block contains 4 words. For each of the following cache configurations,
    a. direct mapped
    b. 2-way set associative
    c. 4-way set associative
    d. fully associative
    specify how the 32-bit address would be partitioned. For example, for a direct mapped cache, you would need to specify which bits are used to select the cache entry and which bits are used to compare against the tag stored in the cache entry.
24. Cache misses can be characterized as one of the following: compulsory misses, capacity misses, and conflict misses. Describe how each of these kinds of misses can be addressed in the hardware.
25. Why virtual memory is called virtual ? What are the different address spaces ? Explain with example how logical address is converted into physical address and also explain how page

replacements take place. Explain the instruction cycle with a neat diagram. Explain the disadvantages of stored program computer.

26. A three-level memory system having cache access time of 5 nsec and disk access time of 40 nsec, has a cache hit ratio of 0.96 and main memory hit ratio of 0.9. What should be the main memory access time to achieve an overall access time of 16 nsec ?
27. According to the following information, determine size of the subfields (in bits) in the address for Direct Mapping and Set Associative Mapping cache schemes :
    We have 256 MB main memory and 1 MB cache memory
    The address space of the processor is 256 MB
    The block size is 128 bytes
    There are 8 blocks in a cache set.

Unit 10 :-

1. What are the functions of I/O interface?
2. How does the processor handle an interrupt request?
3. What are the necessary operations needed to start an I/O operation using DMA?
4. What is the advantage of using interrupt initiated data transfer?
5. Why do you need DMA?
6. What is the difference between subroutine and interrupt service routine?
7. Why I/O devices cannot be directly be connected to the system bus?
8. What is polling?
9. State the differences between memory mapped I/O and I/O mapped I/O.
10. Explain the functions to be performed by a typical I/O interface with a typical input output interface.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Advanced OOPs using C++ Lab
Course Code: CS391
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:
The course presents C++ programming including: advanced C++ environment, exception handling, conception of different file handling, template, STL that aims to:

- Be able to code using more advanced C++ features such as class, objects, operator overloads, dynamic memory allocation, inheritance and polymorphism, exception handling, etc.
- Be able to build class template, function template and also they will able to know how practically STL are works.
- Be able to understand practically different string operations and different file operations, like text file, binary file.

Learning Outcomes:

- Be able to develop different types of computer programs using C++.
- Understand exception handling mechanism and different file (text, binary) operations.
- Understand the usage of template: class template & function template and STL.
- Be able to do different operations on string in C++ programming.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1: Introduction, Basics of C++, Inline function, friend function, function and overloading, inheritance
Exercise No. 2: Exception Handing: throwing, catching, rethrowing mechanism; Multiple catch statement
Exercise No. 3: Template: Class template, Function template
Exercise No. 4: Console I/O operations: C++ streams; C++ stream classes; Unformatted I/O operations; Formatted I/O operations; Managing output with Manipulators.
Exercise No. 5: Working with Files: Text File: Basic file operations on text file: Creating/Writing text into file; Binary File: Creation of file, writing data into file, searching.
Exercise No. 6: Standard Template Library: Components of STL; Containers, Iterator; Applications of container classes.
Exercise No. 7: String Manipulation: The String class; Creating String object; Manipulating strings; Relational operations on strings; String comparison characteristics.

Text Books:

1. Schildt, H., The Complete Reference C++, Tata McGraw Hill Education Pvt. Ltd.
2. E.Balagurusamy; Object Oriented programming with C++; Tata McGraw Hill Education Pvt. Ltd.

Reference Books:
3. Debasish Jana, C++ object oriented programming paradigm, PHI.
4. D. Ravichandran, Programming with C++, Tata McGraw Hill Education Pvt. Ltd.
5. Y.I. Shah and M.H. Thaker, Programming In C++, ISTE/EXCEL BOOKS.

Recommended Systems/Software Requirements:
1. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
2. Turbo C++compiler in Windows XP/7 or Linux Operating System.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

Experiment No 1:Inline Function

Aim: Write a C++ program to find the largest of three numbers using inline function.

Description:
Inline function is one of the important feature of C++. When the program executes the function call instruction the CPU stores the memory address of the instruction following the function call, copies the arguments of the function on the stack and finally transfers control to the specified function. The CPU then executes the function code, stores the function return value in a predefined memory location/register and returns control to the calling function.C++ provides an inline functions to reduce the function call overhead. Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call. This substitution is performed by the C++ compiler at compile time.

Algorithm:
Step 1: Start the program.
Step 2: Declare and define the function largest() as inline.
Step 3: Compare with other variables.
Step 4: Return largest number.
Step 5: Stop the program.

/* Program */

```
#include<iostream.h>
inline int largest(int&a,int&b,int&c)
{
int big=0;
 if(a>b)
big=a;
 else
big=b;
 if(c>big)
 big=c;
 return big;
}
int main()
{
inta,b,c;
cout<<"Enter Three Numbers To Find The Largest "<<endl;
cout<<"a = ";
cin>>a;
cout<<"\nb = ";
cin>>b;
cout<<"\nc = ";
cin>>c;
int large=largest(a,b,c);
cout<<"\n Largest of "<<a<<","<<b<<" and "<<c<<" is "<<large;
getch();
 return(0);
}
```

INPUT 1:

```
Enter Three Numbers To Find The Largest
a = 24
b = 45
c = 23
```

OUTPUT 1:
Largest of 24,45 and 23 is 45

INPUT 2:
Enter Three Numbers To Find The Largest
a = 22
b = 34
c = 56

OUTPUT2:
Largest of 22,34 and 56 is 56

Experiment No. 2:Concept of Class and Object
Aim: Create a class called 'EMPLOYEE' that has
- EMPCODE and EMPNAME as data members
- member function getdata( ) to input data
- member function display( ) to output data
Write a main function to create EMP, an array of EMPLOYEE objects. Accept anddisplay the details of at least 6 employees.

Description:
A class is used to specify the form of an object and it combines data representation and methods for manipulating that data into one neat package. The data and functions within a class are called members of the class.A class definition starts with the keyword class followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations.The keyword public determines the access attributes of the members of the class that follow it. A public member can be accessed from outside the class anywhere within the scope of the class object. You can also specify the members of a class as private or protected which we will discuss in a sub-section.A class provides the blueprints for objects, so basically an object is created from a class. We declare objects of a class with exactly the same sort of declaration that we declare variables of basic types.
Here "EMPLOYEE" is the class, EMPCODE and EMPNAME are the data member. Here getdata() function is used to get input and display() is to show output.

Algorithm:

STEP 1: Start the program.
STEP 2: Declare the class Employee.
STEP 3: empcode and empname are the data members
STEP 4: Declare and define getdata() function to take input from user.
STEP 5: Display() function shows the output.
STEP 6: Array of objects of the class Employee is declared in main() function.
STEP 7: By using Emp[i], access the class members.
STEP 10: Stop the program.

/* Program */

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
class Employee
{
 private: intempcode;
 char empname[10];
 public: void getdata();
 void display();
```

```
};
void Employee::getdata()
{
cout<<"\nNAME :";
cin>>empname;
cout<<"\nCODE :";
cin>>empcode;
}
void Employee::display()
{
cout<<endl<<setw(20)<<empname<<setw(10)<<empcode;
}
int main()
{
 Employee Emp[6];
clrscr();
cout<< "Enter employee details:\n ";
 for(inti=0;i<6;i++)
 {
cout<<"\nemployee "<<i+1<<endl;
Emp[i].getdata();
 }
cout<<"\nEmployee details are as follows :";
 cout<<"\n\n"<<setw(20)<<"NAME"<<setw(10)<<setiosflags(ios::right)<<"CODE";
cout<<"\n------------------------------";
 for(i=0;i<6;i++)
Emp[i].display();
getch();
 return(0);
}
```

INPUT 1:

```
Enter employee details:
employee 1
NAME :ashok
CODE :111
employee 2
NAME :annapurna
CODE :112
employee 3
NAME :anupama
CODE :113
employee 4
NAME :anuradha
CODE :114
employee 5
NAME :ashraya
CODE :115
employee 6
NAME :akash
CODE :116
```

OUTPUT 1:

Employee details are as follows :

NAME	CODE
Ashok	111
Annapurna	112
anupama	113
anuradha	114
ashraya	115
akash	116

INPUT 2:

Enter employee details:
employee 1
NAME :ram
CODE :111
employee 2
NAME :shyam
CODE :112
employee 3
NAME :jodu
CODE :113
employee 4
NAME :madhu
CODE :114
employee 5
NAME :sri
CODE :115
employee 6
NAME :teja
CODE :116

OUTPUT 2:

Employee details are as follows:

NAME	CODE
ram	111
shyam	112
jodu	113
madhu	114
sri	115
teja	116

Experiment No 3:Friend Function

Aim: Create a class 'COMPLEX' to hold a complex number. Write a friend function toadd two complex numbers. Write a main function to add two COMPLEX objects.

Description:

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.Here by using friend function two complex objects are added.

Algorithm:

STEP 1: Start the program.
STEP 2: Declare the class complex.
STEP 3: Declare get_complex(), show_complex(). Also declare add_complex() as friend function.
STEP 4: Define get_complex() to enter the input.
STEP 5: Define show_complex() to show the output.
STEP 6: add_complex() is defined to add two complex number.
STEP 7: c1, c2, c3 are the objects of class complex.
STEP 8: By using the above object member function of the class complex is called.
STEP 10: Stop the program.

/* Program */

```
#include<iostream>
Using namespace std;
class complex
{
 float real,imag;
 public: void get_complex();
 void show_complex();
 friend complex add_complex(complex c1,complex c2);
};
void complex::get_complex()
{
cout<<"Enter real number :";
cin>> real;
cout<<"Enter Imaginary number :";
cin>>imag;
}
void complex::show_complex()
{
cout<<real<<"+i"<<imag;
}
complex add_complex(complex c1,complex c2)
{
 complex c;
c.real=c1.real+c2.real;
c.imag=c1.imag+c2.imag;
 return c;
}
int main()
{
clrscr();
 complex c1,c2,c3;
 c1.get_complex();
 c2.get_complex();
 c3=add_complex(c1,c2);
cout<<"\nComplex Number 1 = ";
 c1.show_complex();
cout<<"\nComplex Number 2 = ";
 c2.show_complex();
cout<<"\nSum of Complex Number 1 and 2 = ";
 c3.show_complex();
getch();
 return 0;
}
```

INPUT 1:
Enter real number:12
Enter Imaginary number:10
Enter real number:3
Enter Imaginary number:5

OUTPUT 1:
Complex Number 1 = 12+i10
Complex Number 2 = 3+i5
Sum of Complex Number 1 and 2 = 15+i15

INPUT 2:
Enter real number:112
Enter Imaginary number:110
Enter real number:13
Enter Imaginary number:15

OUTPUT2:
Complex Number 1 = 112+i110
Complex Number 12 = 13+i15
Sum of Complex Number 1 and 2 = 125+i125

Experiment No. 4: OperatorOverloading
Aim: Create a 'MATRIX' class of size m X n. Overload the '+' operator to add two MATRIX objects. Write a main function to implement it.

Description:
Operator overloading is an important concept in C++. It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on user-defined data type. For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String(concatenation) etc. Almost any operator can be overloaded in C++. However there are few operator which can not be overloaded. Operator that are not overloaded are follows
scope operator - ::
sizeof
member selector - .
member pointer selector - *
ternary operator - ?:
Here addition of two matrix of size m X n are added using '+' operator. So, here '+' operator is overloaded.

Algorithm:
STEP 1: Start the program.
STEP 2: Declare the class mat.
STEP 3: Define the operator '+'.
STEP 4: The function readmat() is defined to insert the elements of the matrix.
STEP 5: The function display() is defined for displaying the outputs.
STEP 6: Objects of mat class is declared in the main() function.
STEP 7: By using that the desired operations are done.
STEP 10: Stop the program.

```
/* Program */
#include<iostream.h>
#include<conio.h>
class mat
{
intm,n,a[20][20];
 public:
 mat(intx,int y);
 void readmat();
 mat operator +(mat);
 void display();
};
 mat :: mat(intx,int y)
{
 m=x;n=y;
 for(inti=0;i<m;i++)
 {
 for(int j=0;j<n;j++)
 a[i][j]=0;
 }
}
 void mat :: readmat()
 {
cout<<"\nenter matrix elements\n";
 for(inti=0;i<m;i++)
 for(int j=0;j<n;j++)
cin>>a[i][j];
 }
 mat mat:: operator +(mat obj)
 {
 mat temp(m,n);
 for(inti=0;i<m;i++)
 for(int j=0;j<n;j++)
 {
temp.a[i][j]=a[i][j]+obj.a[i][j];
 }
 return temp;
 }
 void mat:: display()
 {
inti,j;
 for(i=0;i<m;i++)
 {
cout<<"\n\n";
 for(j=0;j<n;j++)
cout<<"\t"<<a[i][j];
 }
 }
int main()
 {
int m1,n1;
clrscr();
cout<<"\nEnter the size(m,n) of matrix: ";
cin>>m1>>n1;
 mat a(m1,n1),b(m1,n1),c(m1,n1);
cout<<"\nEntermartix 1: ";
```

```
a.readmat();
cout<<"\nEnter matrix 2: ";
b.readmat();
 c=a.operator +(b);
cout<<"\nFirst Matrix :\n";
a.display();
cout<<"\nSecond Matrix :\n";
b.display();
cout<<"\nmatrix 1+matrix 2: ";
c.display();
getch();
 return 0;
 }
```

INPUT 1:

```
Enter the size(m,n) of matrix: 2 2
Enter martix 1: enter matrix elements
3 3
3 3
Enter matrix 2: enter matrix elements
4 4
4 4
```

OUTPUT 1:

```
First Matrix :
3 3
3 3
Second Matrix :
4 4
4 4
matrix 1 + matrix 2:
7 7
7 7
```

INPUT 2:

```
Enter the size(m,n) of matrix: 2 2
Enter martix 1: enter matrix elements
55
5 5
Enter matrix 2: enter matrix elements
4 4
4 4
```

OUTPUT2:

```
First Matrix :
55
5 5
Second Matrix :
4 4
4 4
matrix 1 + matrix 2:
99
9 9
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No. 5: Function Overloading

Aim: Simple Program for Function Overloading Using C++ Programming to calculate the area of circle, rectangle and triangle using function overloading.

Description:

If any class have multiple functions with same names but different parameters then they are said to be overloaded. Function overloading allows you to use the same name for different functions, to perform, either same or different functions in the same class. Function overloading is usually used to enhance the readability of the program. If you have to perform one single operation but with different number or types of arguments, then you can simply overload the function. There are two ways to overload a function 1. By changing number of Arguments 2. By having different types of argument.

Algorithm:

STEP 1: Start the program.
STEP 2: Declare the class name as fn with data members and member functions.
STEP 3: Read the choice from the user.
STEP 4: Choice=1 then go to the step 5.
STEP 5: The function area() to find area of circle with one integer argument.
STEP 6: Choice=2 then go to the step 7.
STEP 7: The function area() to find area of rectangle with two integer argument.
STEP 8: Choice=3 then go to the step 9.
STEP 9: The function area() to find area of triangle with three arguments, two as Integer and one as float.
STEP 10: Choice=4 then stop the program.

/*program*/

```
#include<iostream.h>
#include<stdlib.h>
#include<conio.h>
#define pi 3.14
class fn
{
    public:
      void area(int);  //circle
      void area(int,int);  //rectangle
      void area(float ,int,int);  //triangle
};


void fn::area(int a)
{


cout<<"Area of Circle:"<<pi*a*a;
}
void fn::area(inta,int b)
{
cout<<"Area of rectangle:"<<a*b;
}
void fn::area(float t,inta,int b)
{
cout<<"Area of triangle:"<<t*a*b;
}
```

```
void main()
{
intch;
inta,b,r;
clrscr();
fnobj;
cout<<"\n\t\tFunction Overloading";
cout<<"\n1.Area of Circle\n2.Area of Rectangle\n3.Area of Triangle\n4.Exit\n:”;
cout<<”Enter your Choice:";
cin>>ch;

    switch(ch)
    {
          case 1:
cout<<"Enter Radious of the Circle:";
cin>>r;
obj.area(r);
            break;
          case 2:
cout<<"Enter Sides of the Rectangle:";
cin>>a>>b;
obj.area(a,b);
            break;
          case 3:
cout<<"Enter Sides of the Triangle:";
cin>>a>>b;
obj.area(0.5,a,b);
            break;
          case 4:
            exit(0);
    }
getch();
}
```

OUTPUT:

```
Function Overloading
1. Area of Circle
2. Area of Rectangle
3. Area of Triangle
4. Exit
Enter Your Choice: 2

Enter the Sides of the Rectangle: 5 5

Area of Rectangle is: 25

1. Area of Circle
2. Area of Rectangle
3. Area of Triangle
4. Exit
Enter Your Choice: 4
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No. 6: Inheritance
Aim: To find out the student details using multiple inheritance.

Description:
Multiple inheritance is a feature of some object-oriented computer programming languages in which an object or class can inherit characteristics and features from more than one parent object or parent class. It is distinct from single inheritance, where an object or class may only inherit from one particular object or class.

Algorithm:
Step 1: Start the program.
Step 2: Declare the base class student.
Step 3: Declare and define the function get() to get the student details.
Step 4: Declare the other class sports.
Step 5: Declare and define the function getsm() to read the sports mark.
Step 6: Create the class statement derived from student and sports.
Step 7: Declare and define the function display() to find out the total and average.
Step 8: Declare the derived class object,call the functions get(),getsm() and display().
Step 9: Stop the program.

/*program*/

```
#include<iostream.h>
using namespace std;
class student
{
   protected:
int rno,m1,m2;
   public:
             void get()
            {
cout<<"Enter the Roll no :";
cin>>rno;
cout<<"Enter the two marks   :";
cin>>m1>>m2;
            }
};
class sports
{
   protected:
intsm;                  // sm = Sports mark
   public:
             void getsm()
            {
cout<<"\nEnter the sports mark :";
cin>>sm;

            }
};
class statement:publicstudent,public sports
{
inttot,avg;
   public:
   void display()
            {
              tot=(m1+m2+sm);
avg=tot/3;
```

```
cout<<"\n\n\tRoll No    : "<<rno<<"\n\tTotal      : "<<tot;
cout<<"\n\tAverage    : "<<avg;
          }
};
int main()
{
clrscr();
  statement obj;
obj.get();
obj.getsm();
obj.display();
}
```

INPUT:
Enter the Roll no: 100

Enter two marks
90
80

Enter the Sports Mark: 90

OUTPUT:

Roll No: 100
Total : 260
Average: 86.66

Experiment No. 7:Exception Handling
Aim: Write a C++ program illustrating Exception Handling.

Description:
An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw.

Algorithm:
Step1: Start
Step2: Divide a number by 0 within try block
Step3: Throw the exception
Step4: In catch block display a massage
Step5: stop

/* Program */

```
#include<iostream>
Using namespace std;
int main()
{
inta,b;
cout<<"enter values of a and b \n";
cin>>a;
cin>>b;
int x=a-b;
try
```

```
{
If(x!=0)
{
Cout<<"result is "<<a/x<<"\n";
}
Else
{
Throw(x);
}
}
Catch(inti)
{
Cout<<"Exception caught: x ="<<x<<"\n";
}
```

INPUT 1:
enter values of a and b
20 15

OUTPUT 1:
result is 4

INPUT 2:
enter values of a and b
10 10

OUTPUT2:
Exception caught: x=0

Experiment No. 8: Exception Handling
Aim: To perform exception handling with multiple catch.

Description:
Exceptions can be thrown anywhere within a code block using throw statements. The operand of the throw statements determines a type for the exception and can be any expression and the type of the result of the expression determines the type of exception thrown.The catch block following the try block catches any exception. You can specify what type of exception you want to catch and this is determined by the exception declaration that appears in parentheses following the keyword catch.

Algorithm:
Step 1: Start the program.
Step 2: Declare and define the function test().
Step 3: Within the try block check whether the value is greater than zero or not.
a. if the value greater than zero throw the value and catch the corresponding exception.
b. Otherwise throw the character and catch the corresponding exception.
Step 4: Read the integer and character values for the function test().
Step 5: Stop the program.

/* Program */

```
#include<iostream.h>
#include<conio.h>
void test(int x)
{
  try
  {
```

```
        if(x>0)
          throw x;
    else
          throw 'x';
  }

  catch(int x)
  {
cout<<"Catch a integer and that integer is:"<<x;
  }

  catch(char x)
  {
cout<<"Catch a character and that character is:"<<x;
  }
}

void main()
{
clrscr();
cout<<"Testing multiple catches\n:";
  test(10);
  test(0);
getch();
}
```

OUTPUT:

Testing multiple catches
Catch a integer and that integer is: 10
Catch a character and that character is: x

Experiment No. 9:Function Template
Aim: Write a C++ program to swap the numbers using the concept of function template.

Description:
Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

Syntax of function template:

```
template <class type> ret-type func-name(parameter list)
 {
   // body of function
}
```

Algorithm:
STEP 1: Start the program.
STEP 2: Declare the template class.
STEP 3: Declare and define the functions to swap the values.
STEP 4: Declare and define the functions to get the values.
STEP 5: Read the values and call the corresponding functions.
STEP6: Display the results.
STEP 7: Stop the program.

```
/* Program */
#include<iostream.h>
#include<conio.h>

template<class t>

void swap(t &x,t&y)
{
  t temp=x;
  x=y;
  y=temp;
}

void fun(inta,intb,floatc,float d)
{
cout<<"\na and b before swaping :"<<a<<"\t"<<b;
  swap(a,b);
cout<<"\na and b after swaping  :"<<a<<"\t"<<b;
cout<<"\n\nc and d before swaping :"<<c<<"\t"<<d;
  swap(c,d);
cout<<"\nc and d after swaping  :"<<c<<"\t"<<d;
}

void main()
{
inta,b;
   float c,d;
clrscr();
cout<<"Enter A,B values(integer):";
cin>>a>>b;
cout<<"Enter C,D values(float):";
cin>>c>>d;
   fun(a,b,c,d);
getch();
}
```

INPUT 1:
Enter A, B values (integer): 10 20
Enter C, D values (float): 2.50 10.80

OUTPUT 1:
A and B before swapping: 10 20
A and B after swapping: 20 10

C and D before swapping: 2.50 10.80
C and D after swapping: 10.80 2.50

Experiment No. 10: Function template.
Aim: Program to display largest among two numbers using function templates.

Description:
A function template Large() is defined that accepts two arguments n1 and n2 of data type T. T signifies that argument can be of any data type.Large() function returns the largest among the two arguments using a simple conditional operation.Inside the main() function, variables of three different data types: int, float and char are declared. The variables are then passed to the Large()

function template as normal functions.During run-time, when an integer is passed to the template function, compiler knows it has to generate a Large() function to accept the int arguments and does so.Similarly, when floating-point data and char data are passed, it knows the argument data types and generates the Large() function accordingly.This way, using only a single function template replaced three identical normal functions and made your code maintainable.

Algorithm:
STEP 1: Start the program.
STEP 2: Declare the template function.
STEP 3: Declare and define the function Large() tofind the largest value.
STEP 4: After taking input from user, the largest value is identified
STEP 5: Display the results.
STEP 7: Stop the program.

/*Program*/

```
#include <iostream>
using namespace std;
// template function
template <class T>
T Large(T n1, T n2)
{
        return (n1 > n2) ? n1 : n2;
}
int main()
{
        int i1, i2;
        float f1, f2;
        char c1, c2;

        cout<< "Enter two integers:\n";
        cin>> i1 >> i2;
        cout<< Large(i1, i2) <<" is larger." <<endl;

        cout<< "\nEnter two floating-point numbers:\n";
        cin>> f1 >> f2;
        cout<< Large(f1, f2) <<" is larger." <<endl;
        cout<< "\nEnter two characters:\n";
        cin>> c1 >> c2;
        cout<< Large(c1, c2) << " has larger ASCII value.";
        return 0;
}
```

INPUT 1:
Enter two integers:
5
10
OUTPUT 1:
10 is larger.

INPUT 2:
Enter two floating-point numbers:
12.4
10.2
OUTPUT 2:
12.4 is larger.

INPUT 3:
Enter two characters:
z
Z
OUTPUT 3:
z has larger ASCII value.

Experiment No. 11: Class template
Aim: Simple calculator using Class template. Program to add, subtract, multiply and divide two numbers using class template.

Description:
The class contains two private members of type T: num1 & num2, and a constructor to initalize the members.It also contains public member functions to calculate the addition, subtraction, multiplication and division of the numbers which return the value of data type defined by the user. Likewise, a function displayResult() to display the final output to the screen.In the main() function, two different Calculator objects intCalc and floatCalc are created for data types: int and float respectively. The values are initialized using the constructor.
We use <int> and <float> while creating the objects. These tell the compiler the data type used for the class creation.This creates a class definition each for int and float, which are then used accordingly.Then, displayResult() of both objects is called which performs the Calculator operations and displays the output.

Algorithm:
STEP 1: Start the program.
STEP 2: Declare the template class.
STEP 3: Declare and define the function displayResult() tofind the result of the mathematical operations.
STEP 4: intCalc(2, 1) doing the operations on integer values.
STEP 5: floatCalc(2.4, 1.2) doing the operations on float values.
STEP 6: Display the results.
STEP 7: Stop the program.

/*Program*/
```
#include <iostream>
using namespace std;
template <class T>
class Calculator
{
private:
        T num1, num2;

public:
        Calculator(T n1, T n2)
        {
                num1 = n1;
                num2 = n2;
        }
                void displayResult()
        {
                cout<< "Numbers are: " << num1 << " and " << num2 << "." <<endl;
                cout<< "Addition is: " << add() <<endl;
                cout<< "Subtraction is: " << subtract() <<endl;
                cout<< "Product is: " << multiply() <<endl;
                cout<< "Division is: " << divide() <<endl;
        }
```

```
        T add() { return num1 + num2; }

        T subtract() { return num1 - num2; }

        T multiply() { return num1 * num2; }

        T divide() { return num1 / num2; }
};

int main()
{
        Calculator<int>intCalc(2, 1);
        Calculator<float>floatCalc(2.4, 1.2);

        cout<< "Int results:" <<endl;
        intCalc.displayResult();

        cout<<endl<< "Float results:" <<endl;
        floatCalc.displayResult();

        return 0;
}
```

OUTPUT :

```
Int results:
Numbers are: 2 and 1.
Addition is: 3
Subtraction is: 1
Product is: 2
Division is: 2

Float results:
Numbers are: 2.4 and 1.2.
Addition is: 3.6
Subtraction is: 1.2
Product is: 2.88
Division is: 2
```

Experiment No. 12:CONSOLE INPUT OUTPUT

Aim: Write a C++ program where user will input the text and how many characters are entered would be shown in the output.

Description:

The classes istream and ostream define two member function put() and get() respectively to handle the single character input/output operations. When we type a line of input, the text is sent to the program as soon as we press the RETURN key. The program then reads one character at a time using the statement cin.get(c); and displays it using the statement cout.put(c); The process is terminated when the newline character is encountered.

Algorithm:

STEP 1: Start the program.
STEP 2: Taking inputs from user.
STEP 3: Input will be taken character wise.
STEP 4: After taking input from user, it will count the number of characters.
STEP 5: Display the number of characters.

STEP 6: Stop the program.

```
/*Program*/
#include<iostream>
 using namespace std;
int main()
 {
        int count=0;
        char c;
        cout<<"input text:";
        cin.get(c);
        while(c!='\n')
        {
                cout.put(c);
                count++;
                cin.get(c);
        }
        cout<<"\n number of characters= "<<count;
        return 0;
 }
```

INPUT 1:
input text: object oriented programming

OUTPUT 1:
object oriented programming
number of characters=27

INPUT 1:
input text: happy new year
OUTPUT2:
happy new year
number of characters=14

Experiment No. 13: CONSOLE INPUT OUTPUT
Aim: Write a C++ program to implement the working principle ofgetline() function.

Description:
We can read and display a line of text more efficiently using the line oriented input functions getline(). The getline() function reads a whole line of a text that ends with a newline character. The reading is terminated as soon as either the newline character '\n' is encountered or size-1 characters are read (whichever occurs first). The newline character is read byt not saved. Instead it is replaced by NULL character.

Algorithm:
STEP 1: Start the program.
STEP 2: Taking inputs from user.
STEP 3: Input will be taken as a whole line.
STEP 4: After taking input from user, it will print the desired output.
STEP 5: Display the result.
STEP 6: Stop the program.

```
/*Program*/
#include<iostream>
using namespace std;
int main()
{
        int size=20;
        char city[20];
        cout<<"enter the city name: \n";
        cin>>city;
        cout<<"city name: "<<city<<"\n\n";
        cout<<"enter the city name again: \n";
        cin.getline(city,size);
        cout<<"city name now: "<<city<<"\n\n";
        cout<<"enter the another city name: \n";
        cin.getline(city,size);
        cout<<"New city name is: "<<city<<"\n\n";
}
```

INPUT 1:
Enter city name:
Delhi

OUTPUT 1:
City name:
Delhi
City name: Delhi
enter the city name again:
city name now:
enter the another city name:
chennai
New city name is: Chennai

INPUT 1:
Enter city name:
New Delhi

OUTPUT 1:
City name: New
enter the city name again:
city name now: Delhi
enter the another city name:
Greater Kolkata
New city name is: Greater Kolkata

Experiment No. 14:CONSOLE INPUT OUTPUT
Aim: Write a C++ program to create the following format.

ITEMS	COST	TOTAL VALUE
10	75	750
6	100	600
12	60	720
15	99	1485

GRAND TOTAL= 3555

Description:
For formatting the output width() function is used. To set the required field width we use it. The output will be displayed in given width.

Algorithm:

STEP 1: Start the program.
STEP 2: Define arrays of items and cost.
STEP 3: Set the output with width() function
STEP 4: Display the desired output.
STEP 5: Stop the program.

/*Program*/

```
#include<iostream>
using namespace std;
int main()
{
        int items[4]={10,6,12,15};
        int cost[4]={75,100,60,99};
        cout.width(5);
        cout<<"ITEMS";
        cout.width(8);
        cout<<"COST";
        cout.width(15);
        cout<<"TOTAL VALUE"<<"\n";
        int sum=0;
        for(inti=0;i<4;i++)
        {
                cout.width(5);
                cout<<items[i];
                cout.width(8);
                cout<<cost[i];
                int value=items[i] * cost[i];
                cout.width(15);
                cout<<value<<"\n";
                sum=sum+value;
        }
        cout<<"\n GRAND TOTAL= ";
        cout.width(2);
        cout<<sum<<"\n";
        return 0;
}
```

OUTPUT 1:

ITEMS	COST	TOTAL VALUE
10	75	750
6	100	600
12	60	720
15	99	1485

GRAND TOTAL= 3555

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No. 15:CONSOLE INPUT OUTPUT

Aim: Write a C++ program where you find the square root of any five values. Format the desired output with precision() function.

Description:

By using precision () function we just format the output. This function is used to set number of decimal points to a float value.

Algorithm:

STEP 1: Start the program.
STEP 2: Set precision.
STEP 3: Set the width of output and find the square root of particular values.
STEP 4: Display the desired output.
STEP 5: Stop the program.

/*Program*/

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{       cout<<"precision set to 3 digits \n\n";
        cout.precision(3);
        cout.width(10);
        cout<<"VALUE";
        cout.width(15);
        cout<<"SORT_OF_VALUE"<<"\n";
        for(int n=1;n<=5;n++)
        {
                cout.width(8);
                cout<<n;
                cout.width(13);
                cout<<sqrt(n)<<"\n";
        }
        cout<<"\n PRECISION SET TO 5 DIGITS \n\n";
        cout.precision(5);
        cout<<"sqrt(10)= "<<sqrt(10)<<"\n\n";
        cout.precision(0);
        cout<<"sqrt(10)= "<<sqrt(10)<<"\n default settings \n";
        return 0;

}
```

OUTPUT:

precision set to 3 digits

VALUE	SORT_OF_VALUE
1	1
2	1.41
3	1.73
4	2
5	2.24

PRECISION SET TO 5 DIGITS

sqrt(10)= 3.1623
sqrt(10)= 3
default settings

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No. 16:CONSOLE INPUT OUTPUT
Aim: Fill the blank spaces of the above program with '.'

Description:
By using fill () function we just format the output. The blank spaces of the above output is replaced by '.'

Algorithm:
STEP 1: Start the program.
STEP 2: Set precision.
STEP 3: Set the width of output and find the square root of particular values.
STEP 4: Fill up the blank spaces with '.' Using fill() function.
STEP 5: Display the desired output.
STEP 6: Stop the program.

/*Program*/

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
        cout<<"precision set to 3 digits \n\n";
        cout.precision(3);
        cout.fill('.');
        cout.width(10);
        cout<<"VALUE";
        cout.width(15);
        cout<<"SORT_OF_VALUE"<<"\n";
        for(int n=1;n<=5;n++)
        {
                cout.width(8);
                cout<<n;
                cout.width(13);
                cout<<sqrt(n)<<"\n";
        }
        cout<<"\n PRECISION SET TO 5 DIGITS \n\n";
        cout.precision(5);
        cout<<"sqrt(10)= "<<sqrt(10)<<"\n\n";
        cout.precision(0);
        cout<<"sqrt(10)"<<sqrt(10)<<"default setting \n";
        return 0;

}
```

OUTPUT:

```
precision set to 3 digits

    …….VALUE…… SORT_OF_VALUE
……..1……1
……...……...2……...1.41
……..3……...……1.73
……..4……2
……..5……2.24
PRECISION SET TO 5 DIGITS

sqrt(10)= 3.1623
```

sqrt(10)= 3
default settings

Experiment No. 17:FILE HANDLING
Aim: Write a Program for Read File Operation Using C++ Programming

Description:
File handling concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory (Hard disk). fstream is used to both read and write data from/to files

Algorithm:
STEP 1: Start the program.
STEP 2: Declare the variables.
STEP 3: Get the file name to read.
STEP 4: Using ifstreamin(filename) check whether the file exist.
STEP 5: If the file exist then check for the end of file condition.
STEP 6: Read the contents of the file.
STEP 7: Print the contents of the file.
STEP 8: Stop the program.

/*Program*/

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
        char c,fname[10];
cout<<"Enter file name:";
cin>>fname;
ifstream in(fname);
        if(!in)
        {
cout<<"File Does not Exist";
                return;
        }
cout<<"\n\n";
        while(in.eof()==0)
        {
in.get(c);
cout<<c;
        }

}
```

OUTPUT:
Enter File name: one.txt
INDIA

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No. 18:FILE HANDLING

Aim:Write a Program for Read & Write File Operation (Convert lowercase to uppercase) Using C++ Programming

Description:

A file must be opened before you can read from it or write to it. Either the ofstream or fstream object may be used to open a file for writing or ifstream object is used to open a file for reading purpose only.C++ provides a special function, eof( ), that returns nonzero (meaning TRUE) when there are no more data to be read from an input file stream, and zero (meaning FALSE) otherwise.

Algorithm:

STEP 1: Start the program.
STEP 2: Declare the variables.
STEP 3: Read the file name.
STEP 4: open the file to write the contents.
STEP 5: writing the file contents up to reach a particular condition.
STEP6: write the file contents as uppercase.
STEP7: open the file to read the contents.
STEP 8: Stop the program.

/*Program*/

```
#include<fstream.h>
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<iostream.h>
#include<conio.h>
void main()
{           char c,u;
          char fname[10];
clrscr();
ofstream out;
cout<<"Enter File Name:";
cin>>fname;
out.open(fname);
cout<<"Enter the text(Enter # at end)\n";   //write contents to file
          while((c=getchar())!='#')
          {
                    u=c-32;
                    out<<u;
          }
out.close();
ifstream in(fname);       //read the contents of file
cout<<"\n\n\t\tThe File contains\n\n";
          while(in.eof()==0)
          {         in.get(c);
cout<<c;
          }
getch();
 }
```

OUTPUT:

```
Enter File Name: two.txt
Enter contents to store in file (enter # at end)
oops programming
The File Contains
OOPS PROGRAMMING
```

Experiment No. 19:FILE HANDLING
Aim: Program to count number of words in a file.

Description:
File handling concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory. For read and write from a file we need another standard C++ library called fstream. Always test for the end-of-file condition before processing data read from an input file stream. Use a while loop for getting data from an input file stream. A for loop is desirable only when you know the exact number of data items in the file, which we do not know.

Algorithm:
STEP 1: Start the program.
STEP 2: Declare the variables.
STEP 3: Read the file name.
STEP 5: Check the eof condition
STEP6: Count the number of words
STEP 7: Stop the program.

/*Program*/

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
ifstream fin;
fin.open("out.txt");

int count = 0;
   char word[30];

   while(!fin.eof())
   {
      fin >> word;
      count++;
   }

cout<< "Number of words in file are " << count;

fin.close();
   return 0;
}
```

OUTPUT:
Number of words in file are 20.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No. 20:FILE HANDLING

Aim: Program to count number of lines in a text file.

Description:

File handling concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory. For read and write from a file we need another standard C++ library called fstream. Always test for the end-of-file condition before processing data read from an input file stream. Use a while loop for getting data from an input file stream. A for loop is desirable only when you know the exact number of data items in the file, which we do not know.

Algorithm:

STEP 1: Start the program.
STEP 2: Declare the variables.
STEP 3: Read the file name.
STEP 5: Check the eof condition
STEP 6: Count the number of lines
STEP 7: Stop the program.

/*Program*/

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
ifstream fin;
fin.open("out.txt");

int count = 0;
   char str[80];

   while(!fin.eof())
   {
fin.getline(str,80);
      count++;
   }
cout<< "Number of lines in file are " << count;
fin.close();
   return 0;
}
```

OUTPUT:

Number of lines in file are 5.

Experiment No. 21:FILE HANDLING

Aim: Program to implement searching operation on binary file in C++

Description:

When data is stored in a file in the binary format, reading and writingdata is faster because no time is lost in converting the data from one format to another format. Such files are called binary files. This following program explains how to create binary files and also how to read, write, search, delete and modify data from binary files.

Algorithm:

STEP 1: Start the program.
STEP 2: Declare the variables and functions with definitions.
STEP 3: Read the file name.
STEP 5: Check the eof condition
STEP 6: searching is done.
STEP 7: Stop the program.

/*Program*/

```
#include<fstream.h>
#include<conio.h>
#include<stdlib.h>
class student
{
        introllno;
        char name[20];
        char branch[3];
        float marks;
        char grade;
        public:
                void getdata()
                {
                        cout<<"Rollno: ";
                        cin>>rollno;
                        cout<<"Name: ";
                        cin>>name;
                        cout<<"Branch: ";
                        cin>>branch;
                        cout<<"Marks: ";
                        cin>>marks;

                        if(marks>=75)
                        {
                                grade = 'A';
                        }
                        else if(marks>=60)
                        {
                                grade = 'B';
                        }
                        else if(marks>=50)
                        {
                                grade = 'C';
                        }
                        else if(marks>=40)
                        {
                                grade = 'D';
                        }
                        else
```

```
                {
                        grade = 'F';
                }
        }
        void putdata()
        {
                cout<<"Rollno: "<<rollno<<"\tName: "<<name<<"\n";
                cout<<"Marks: "<<marks<<"\tGrade: "<<grade<<"\n";
        }
        intgetrno()
        {
                return rollno;
        }
}stud1;
void main()
{
        clrscr();

        fstreamfio("marks.dat", ios::in | ios::out);
        char ans='y';
        while(ans=='y' || ans=='Y')
        {
                stud1.getdata();
                fio.write((char *)&stud1, sizeof(stud1));
                cout<<"Record added to the file\n";
                cout<<"\nWant to enter more ? (y/n)..";
                cin>>ans;
        }
        clrscr();
        intrno;
        long pos;
        char found='f';
        cout<<"Enter rollno of student to be search for: ";
        cin>>rno;
        fio.seekg(0);
        while(!fio.eof())
        {
                pos=fio.tellg();
                fio.read((char *)&stud1, sizeof(stud1));
                if(stud1.getrno() == rno)
                {
                        stud1.putdata();
                        fio.seekg(pos);
                        found='t';
                        break;
                }
        }
        if(found=='f')
        {
                cout<<"\nRecord not found in the file..!!\n";
                cout<<"Press any key to exit...\n";
                getch();
                exit(2);
        }
fio.close();
getch();
}
```

OUTPUT:
Rollno: 1
Name: Aman
Branch: CSE
Marks: 96
Recorded added to the file

Want to enter more? (y/n)..y
Rollno: 2
Name: Savvy
Branch: IT
Marks: 91
Recorded added to the file

Want to enter more? (y/n)..n

Searching
Enterrollno of student to be search for: 2
Rollno: 2 Name: Savvy
Marks: 91 Grade: A

Experiment No. 22:STL
Aim:C++ Program to Implement Queue in STL.

Description:
The C++ STL (Standard Template Library) is a powerful set of C++ template classes to provide general-purpose templatized classes and functions that implement many popular and commonly used algorithms and data structures like vectors, lists, queues, and stacks.This C++ Program demonstrates implementation of Queue in STL. Here is source code of the C++ Program to demonstrate Queue in STL.

Algorithm:
STEP 1: Start the program.
STEP 2: Declare the variables and functions with definitions.
STEP 3: Define STL of queue
STEP 5: Doing queue operations
STEP 6: Stop the program.

/*Program*/
```
#include <iostream>
#include <queue>
#include <string>
#include <cstdlib>
using namespace std;
int main()
{
   queue<int> q;
int choice, item;
   while (1)
   {
cout<<"\n---------------------"<<endl;
cout<<"Queue Implementation in Stl"<<endl;
cout<<"\n---------------------"<<endl;
cout<<"1.Insert Element into the Queue"<<endl;
cout<<"2.Delete Element from the Queue"<<endl;
```

```
            cout<<"3.Size of the Queue"<<endl;
cout<<"4.Front Element of the Queue"<<endl;
cout<<"5.Last Element of the Queue"<<endl;
cout<<"6.Exit"<<endl;
cout<<"Enter your Choice: ";
cin>>choice;
        switch(choice)
        {
        case 1:
cout<<"Enter value to be inserted: ";
cin>>item;
q.push(item);
            break;
        case 2:
            item = q.front();
q.pop();
cout<<"Element "<<item<<" Deleted"<<endl;
            break;
        case 3:
             cout<<"Size of the Queue: ";
             cout<<q.size()<<endl;
            break;
        case 4:
cout<<"Front Element of the Queue: ";
             cout<<q.front()<<endl;
            break;
        case 5:
cout<<"Back Element of the Queue: ";
cout<<q.back()<<endl;
            break;
        case 6:
            exit(1);
                break;
        default:
cout<<"Wrong Choice"<<endl;
        }
    }
    return 0;
}
```

OUTPUT:

```
1.Insert Element into the Queue
2.Delete Element from the Queue
3.Size of the Queue
4.Front Element of the Queue
5.Last Element of the Queue
6.Exit
Enter your Choice: 1
Enter value to be inserted: 9

---------------------
Queue Implementation in Stl

---------------------
1.Insert Element into the Queue
2.Delete Element from the Queue
3.Size of the Queue
```

```
4.Front Element of the Queue
5.Last Element of the Queue
6.Exit
Enter your Choice: 1
Enter value to be inserted: 8
---------------------
Queue Implementation in Stl

---------------------
1.Insert Element into the Queue
2.Delete Element from the Queue
3.Size of the Queue
4.Front Element of the Queue
5.Last Element of the Queue
6.Exit
Enter your Choice: 3
Size of the Queue: 2
```

Experiment No. 23:STL
Aim:C++ Program to Implement Vector in STL

Description:
Vector is a template class that is a perfect replacement for the good old C-style arrays. It allows the same natural syntax that is used with plain arrays but offers a series of services that free the C++ programmer from taking care of the allocated memory and help operating consistently on the contained objects.This C++ Program demonstrates implementation of Vector in STL.Here is source code of the C++ Program to demonstrate Vector in STL.

Algorithm:
STEP 1: Start the program.
STEP 2: Declare the variables and functions with definitions.
STEP 3: Define STL of vector
STEP 5: Doing vector operations
STEP 6: Stop the program.

/*Program*/

```
#include <iostream>
#include <vector>
#include <string>
#include <cstdlib>
using namespace std;
int main()
{
   vector<int>ss;
   vector<int>::iterator it;
int choice, item;
   while (1)
   {
cout<<"\n---------------------"<<endl;
cout<<"Vector Implementation in Stl"<<endl;
cout<<"\n---------------------"<<endl;
cout<<"1.Insert Element into the Vector"<<endl;
cout<<"2.Delete Last Element of the Vector"<<endl;
```

```
cout<<"3.Size of the Vector"<<endl;
cout<<"4.Display by Index"<<endl;
cout<<"5.Dislplay by Iterator"<<endl;
cout<<"6.Clear the Vector"<<endl;
cout<<"7.Exit"<<endl;
cout<<"Enter your Choice: ";
cin>>choice;
      switch(choice)
      {
      case 1:
cout<<"Enter value to be inserted: ";
cin>>item;
ss.push_back(item);
         break;
      case 2:
cout<<"Delete Last Element Inserted:"<<endl;
ss.pop_back();
         break;
      case 3:
cout<<"Size of Vector: ";
cout<<ss.size()<<endl;
         break;
      case 4:
cout<<"Displaying Vector by Index: ";
         for (inti = 0; i<ss.size(); i++)
         {
cout<<ss[i]<<" ";
         }
cout<<endl;
         break;
      case 5:
cout<<"Displaying Vector by Iterator: ";
         for (it = ss.begin(); it != ss.end(); it++)
         {
cout<<*it<<" ";
         }
cout<<endl;
         break;
      case 6:
ss.clear();
cout<<"Vector Cleared"<<endl;
         break;
      case 7:
         exit(1);
         break;
      default:
cout<<"Wrong Choice"<<endl;
      }
   }
   return 0;
}
```

OUTPUT:

```
----------
Vector Implementation in Stl

----------
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 4

----------
Vector Implementation in Stl

----------
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 6

----------
Vector Implementation in Stl

----------
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 3

----------
Vector Implementation in Stl

----------
1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 8
```

Vector Implementation in Stl

1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 9

Vector Implementation in Stl

1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 1
Enter value to be inserted: 2

Vector Implementation in Stl

1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 3
Size of Vector: 6

Vector Implementation in Stl

1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 4
Displaying Vector by Index: 4 6 3 8 9 2

Vector Implementation in Stl

1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 2
Delete Last Element Inserted:

Vector Implementation in Stl

1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 3
Size of Vector: 5

Vector Implementation in Stl

1.Insert Element into the Vector
2.Delete Last Element of the Vector
3.Size of the Vector
4.Display by Index
5.Dislplay by Iterator
6.Clear the Vector
7.Exit
Enter your Choice: 5
Displaying Vector by Iterator: 4 6 3 8 9

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No. 24:STL
Aim:C++ Program to Implement Set in STL

Description:
This C++ Program demonstrates implementation of Set in STL.Here is source code of the C++ Program to demonstrate Set in STL.

Algorithm:
STEP 1: Start the program.
STEP 2: Declare the variables and functions with definitions.
STEP 3: Define STL of set
STEP 5: Doing set operations
STEP 6: Stop the program.

/*Program*/

```
#include <iostream>
#include <set>
#include <string>
#include <cstdlib>
using namespace std;
int main()
{
   set<int>st;
   set<int>::iterator it;
int choice, item;
   while (1)
   {
cout<<"\n----------------------"<<endl;
cout<<"Set Implementation in Stl"<<endl;
cout<<"\n----------------------"<<endl;
cout<<"1.Insert Element into the Set"<<endl;
cout<<"2.Delete Element of the Set"<<endl;
cout<<"3.Size of the Set"<<endl;
cout<<"4.Find Element in a Set"<<endl;
cout<<"5.Dislplay by Iterator"<<endl;
cout<<"6.Exit"<<endl;
cout<<"Enter your Choice: ";
cin>>choice;
       switch(choice)
       {
       case 1:
cout<<"Enter value to be inserted: ";
cin>>item;
st.insert(item);
          break;
       case 2:
cout<<"Enter the element to be deleted: ";
cin>>item;
st.erase(item);
          break;
       case 3:
cout<<"Size of the Set: ";
cout<<st.size()<<endl;
          break;
       case 4:
           cout<<"Enter the element to be found: ";
           cin>>item;
```

```
            it = st.find(item);
                if (it != st.end())
cout<<"Element "<<*it<<" found in the set" <<endl;
            else
cout<<"No Element Found"<<endl;
            break;
        case 5:
cout<<"Displaying Map by Iterator: ";
            for (it = st.begin(); it != st.end(); it++)
            {
cout<< (*it)<<" ";
            }
cout<<endl;
            break;
        case 6:
            exit(1);
                break;
        default:
cout<<"Wrong Choice"<<endl;
        }
    }
    return 0;
}
```

OUTPUT:

```
----------------------
Set Implementation in Stl
 ----------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 1

----------------------
Set Implementation in Stl
 ----------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 2

----------------------
Set Implementation in Stl
 ----------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
```

```
6.Exit
Enter your Choice: 1
Enter value to be inserted: 3

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 4

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 5

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 4

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 3

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
```

```
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 2

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 1
Enter value to be inserted: 1

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 3
Size of the Set: 5

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 5
Displaying Map by Iterator: 1 2 3 4 5

---------------------
Set Implementation in Stl
 ---------------------
1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 4
Enter the element to be found: 3
Element 3 found in the set
```

Set Implementation in Stl

1.Insert Element into the Set
2.Delete Element of the Set
3.Size of the Set
4.Find Element in a Set
5.Dislplay by Iterator
6.Exit
Enter your Choice: 2
Enter the element to be deleted: 5

Experiment No. 25:String Manipulation
Aim: Write a C++ program to copy a string from another string. Also concatenate them and find the total length after concatenation.

Description:
C++ provides following two types of string representations:
-The C-style character string.
-The string class type introduced with Standard C++.
The C-style character string originated within the C language and continues to be supported within C++. This string is actually a one-dimensional array of characters which is terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.
C++ supports a wide range of functions that manipulate null-terminated strings:
strcpy(s1, s2);
Copies string s2 into string s1.
strcat(s1, s2);
Concatenates string s2 onto the end of string s1.
strlen(s1);
Returns the length of string s1.
strcmp(s1, s2);
Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
strchr(s1, ch);
Returns a pointer to the first occurrence of character ch in string s1.
strstr(s1, s2);
Returns a pointer to the first occurrence of string s2 in string s1.

Algorithm:
STEP 1: Start the program.
STEP 2: Declare the string variables.
STEP 3: Copy the content of str1 to str3
STEP 5: Concatenate str1 and str2
STEP 6: Find the length of str1
STEP 7: Stop the program.

/*Program*/
```
#include <iostream>
#include <cstring>
using namespace std;
int main ()
{
  char str1[10] = "Hello";
  char str2[10] = "World";
  char str3[10];
intlen ;
```

```
  // copy str1 into str3
strcpy( str3, str1);
cout<< "strcpy( str3, str1) : " << str3 <<endl;

  // concatenates str1 and str2
strcat( str1, str2);
cout<< "strcat( str1, str2): " << str1 <<endl;

  // total lenghth of str1 after concatenation
len = strlen(str1);
cout<< "strlen(str1) : " <<len<<endl;

  return 0;
}
```

OUTPUT:

```
strcpy( str3, str1) : Hello
strcat( str1, str2): HelloWorld
strlen(str1) : 10
```

Experiment No. 26:String Manipulation

Aim: Write a C++ program to implement he String Class in C++.

Description:

The standard C++ library provides a string class type that supports all the operations mentioned above, additionally much more functionality.

Algorithm:

STEP 1: Start the program.
STEP 2: Declare the string variables.
STEP 3: Copy str1 into str3
STEP 5: Concatenate str1 and str2 and stores into str3
STEP 6: Find the length of str3
STEP 7: Stop the program.

/*Program*/

```
#include <iostream>
#include <string>
using namespace std;
int main ()
{
  string str1 = "Hello";
  string str2 = "World";
  string str3;
intlen ;

  // copy str1 into str3
  str3 = str1;
cout<< "str3 : " << str3 <<endl;

  // concatenates str1 and str2
  str3 = str1 + str2;
cout<< "str1 + str2 : " << str3 <<endl;

  // total length of str3 after concatenation
```

```
len = str3.size();
cout<< "str3.size() :  " <<len<<endl;

  return 0;
}
```

OUTPUT:

```
str3 : Hello
str1 + str2 : HelloWorld
str3.size() :  10
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Analog & Digital Electronics Lab
Course Code: CS392
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:
The objective of this course is to introduce the organization of a computer and its principal components, viz, ALU, Control, Memory and Input/output. The course will also enable the student to understand the design components of a digital subsystem that required realizing various components such as ALU, Control, etc.

Learning Outcomes:
Upon successful completion of the Lab course, a student will be able to:
1. An ability to implement basic gates and their operations.
2. An ability to understand and implement Flip Flops
3. An ability to understand and implement Multiplexers
4. An ability to understand and implement shift registers and counters
5. An ability to understand and implement Encoders and Decoders
6. An ability to understand and implement Half adder and Full adder Must be able to build a small 8 bit processor that supports reading from memory (16 bytes), Execute 3 instructions, and add/subtract/stop. All operations are to be performed on set of 4 registers. Must implement program counter and decoder to fetch the next instruction.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1: Realization of basic gates using Universal logic gates
Exercise No. 2: Code conversion circuits- BCD to Excess-3
Exercise No. 3: One bit and two bit comparator circuits
Exercise No. 4: Construction of simple Decoder and Multiplexer circuits using NAND gate.
Exercise No. 5: Construction of simple arithmetic circuits-Adder, Subtractor.
Exercise No. 6: Realization of RS-JK and D flip-flops using Universal logic gates.
Exercise No. 7: Realization of Ring counter and Johnson's counter.
Exercise No. 8: Study of Diode as clipper & clamper.
Exercise No. 9: Study of Zener diode as a voltage regulator.
Exercise No. 10: Study of ripple and regulation characteristics of full wave rectifier without and with capacitor filter.
Exercise No. 11: Study of characteristics curves of B.J.T.

Text Book:

Recommended Systems/Software Requirements:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No.: 01

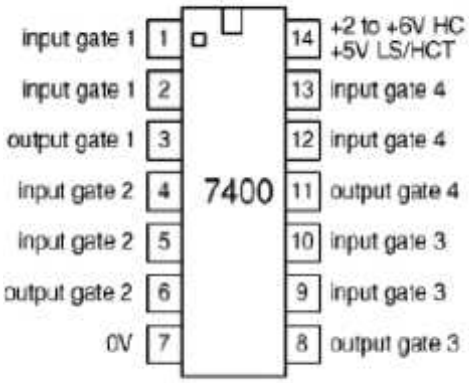
Experiment Name: Realization of basic gates using Universal logic gates

AIM: To construct logic gates NOT, AND, OR, EX-OR, EX-NOR of basic gates using NAND gate and verify their truth tables.

Apparatus Required: Digital trainer kit, patch cords, IC 7400.

Pin Diagram:

IC 7400 NAND gate



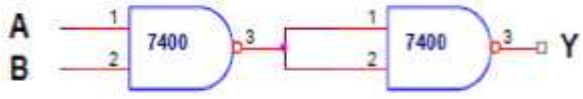
Circuit Diagrams

Not gate

Truth Table

INPUT A	OUTPUT Y
0	1
1	0

AND gate

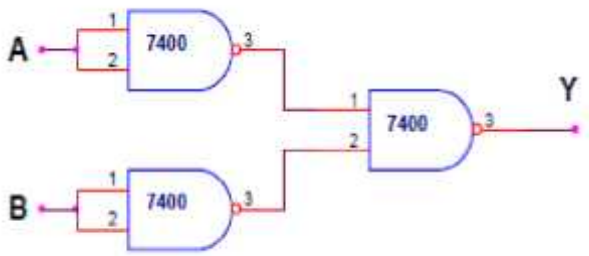


INPUT		OUTPUT Y
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

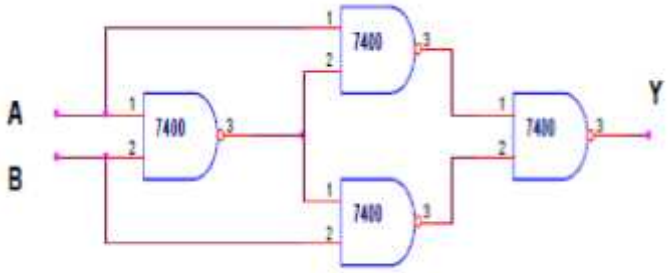
Lab Manual

OR gate



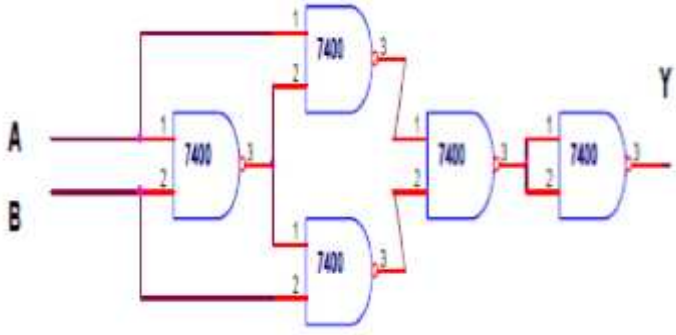
INPUT		OUTPUT Y
A	B	
0	0	0
0	1	1
1	0	1
1	1	1

Ex-OR gate



INPUT		OUTPUT Y
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Ex-NOR gate



INPUT		OUTPUT Y
A	B	
0	0	1
0	1	0
1	0	0
1	1	1

Procedure:

1. Connect the logic gates as shown in the diagrams using IC 7400 NAND gate.
2. Feed the logic signals 0 or 1 from the logic input switches in different combinations at the inputs A & B.
3. Monitor the output using logic output LED indicators.
4. Repeat steps 1 to 3 for NOT, AND, OR, EX – OR & EX-NOR operations and compare the outputs with the truth tables.

Precautions:
1. All the connections should be made properly.
2. IC should not be reversed.

Result: Different logic gates are constructed using NAND gates and their truth tables are verified.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No.: 02

Experiment Name: Code conversion circuits- BCD to Excess-3.

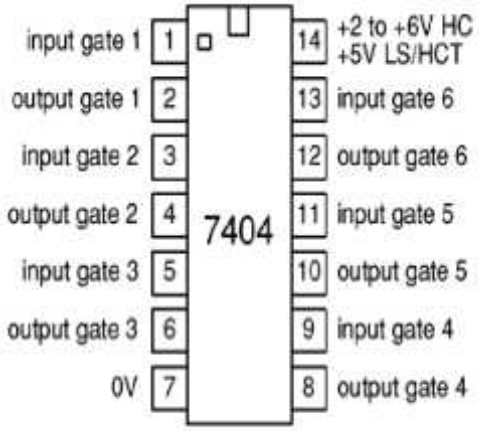
AIM: To design and implement 4 bit BCD to excess-3 converter

Apparatus Required: Digital trainer kit, patch cords, IC 7432, IC 7404, IC 7408, IC 7486.

Pin Diagram:

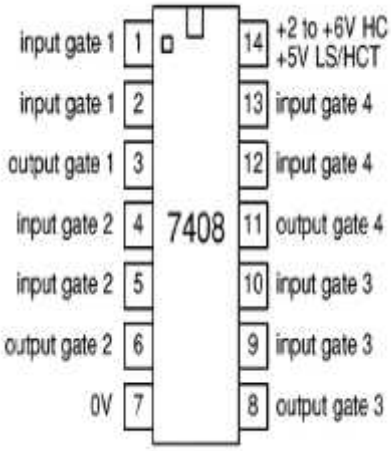
NOT gate

IC 7404



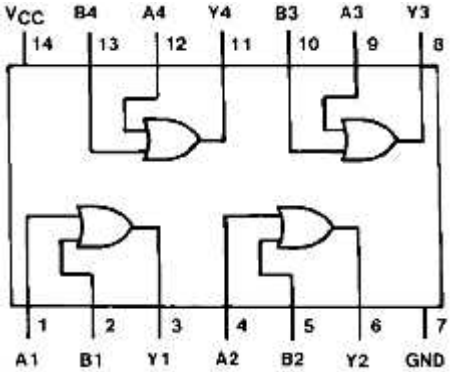
AND gate

IC 7408



OR gate

IC 7432

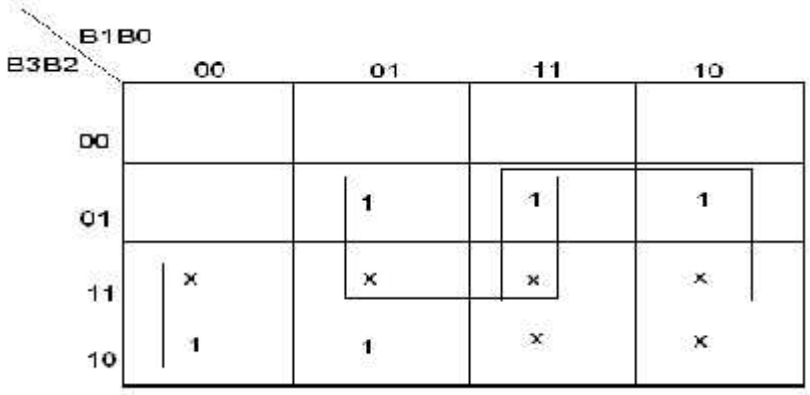


Theory:

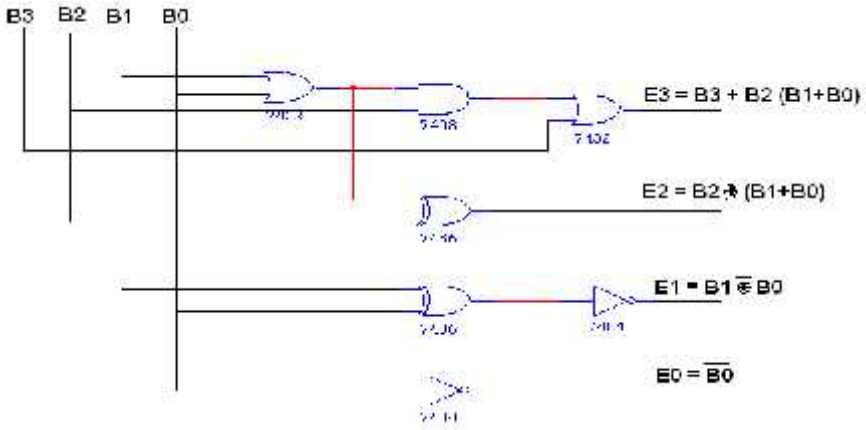
A code converter is a circuit that makes the two systems compatible even though each uses a different binary code. To convert from binary code to Excess-3 code, the input lines must supply the bit combination of elements as specified by code and the output lines generate the corresponding bit combination of code. Each one of the four maps represents one of the four outputs of the circuit as a function of the four input variables.

LOGIC DIAGRAM:

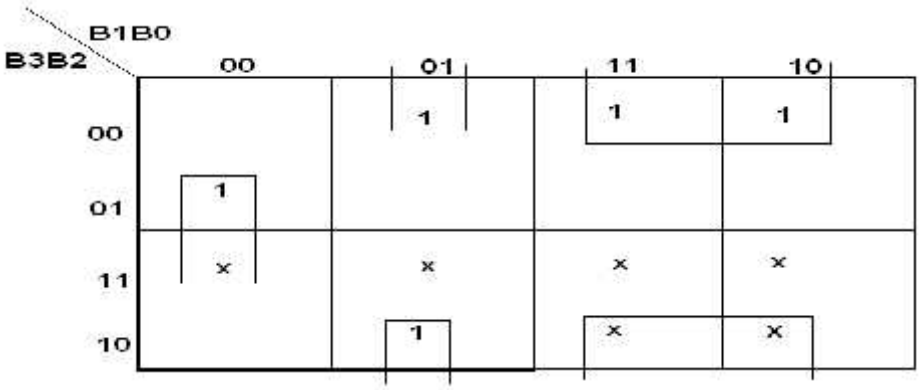
K-map for E3



E3 = B3 + B2 (B0 + B1)

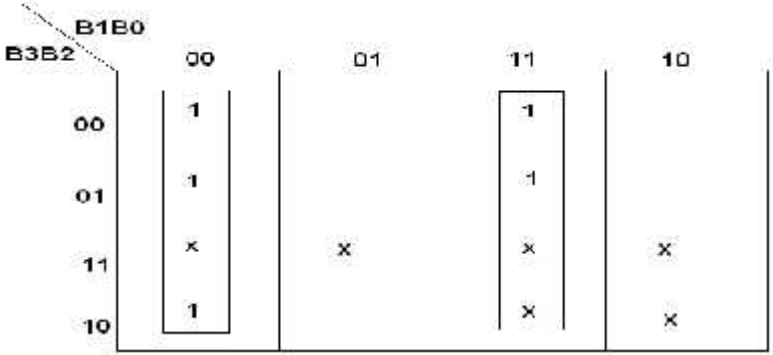


K-map for E2



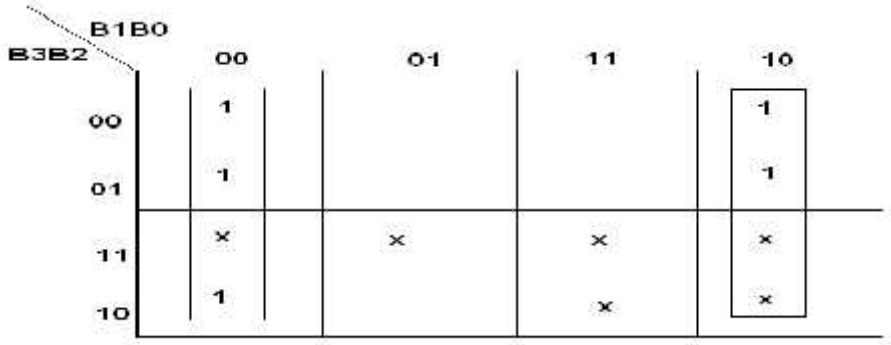
E2 = B2⊕(B1 + B0)

K-map for E1



E1 = B1⊕B0

K-map for E0



TRUTH TABLE:

BCD Input				**Excess-3 Output**			
B3	**B2**	**B1**	**B0**	**E3**	**E2**	**E1**	**E0**
0	**0**	**0**	**0**	**0**	**0**	**1**	**1**
0	**0**	**0**	**1**	**0**	**1**	**0**	**0**
0	**0**	**1**	**0**	**0**	**1**	**0**	**1**
0	**0**	**1**	**1**	**0**	**1**	**1**	**0**
0	**1**	**0**	**0**	**0**	**1**	**1**	**1**
0	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**1**	**0**	**1**	**0**	**0**	**1**
0	**1**	**1**	**1**	**1**	**0**	**1**	**0**
1	**0**	**0**	**0**	**1**	**0**	**1**	**1**
1	**0**	**0**	**1**	**1**	**1**	**0**	**0**
1	**0**	**1**	**0**	**x**	**x**	**x**	**x**
1	**0**	**1**	**1**	**x**	**x**	**x**	**x**
1	**1**	**0**	**0**	**x**	**x**	**x**	**x**
1	**1**	**0**	**1**	**x**	**x**	**x**	**x**
1	**1**	**1**	**0**	**x**	**x**	**x**	**x**
1	**1**	**1**	**1**	**x**	**x**	**x**	**x**

PROCEDURE:

i. Connections were given as per circuit diagram.
ii. Logical inputs were given as per truth table
iii. Observe the logical output and verify with the truth tables.

Result:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

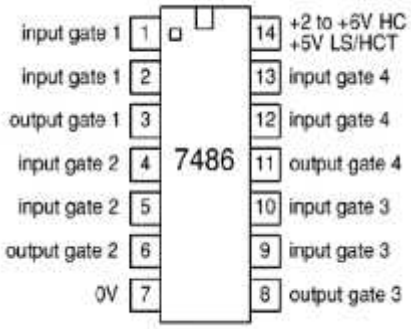
Experiment Name: One bit and two bit comparator circuits.

AIM: To design and implement 1 bit and 2 bit comparators circuits and verify its outputs.

Apparatus Required: Digital trainer kit, patch cords, IC 7404, IC 7408, IC 7486, IC 7432.

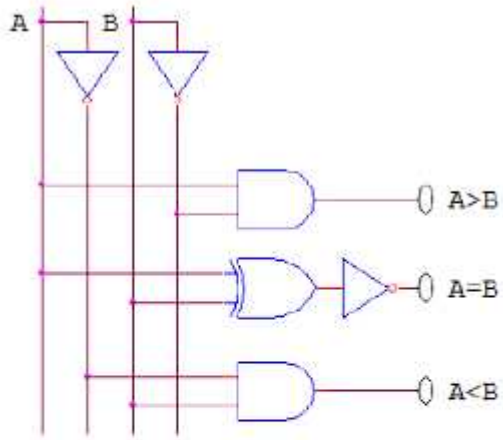
Pin Diagram:

Ex-OR gate



LOGIC DIAGRAM:

1bit Comparator



Truth Table

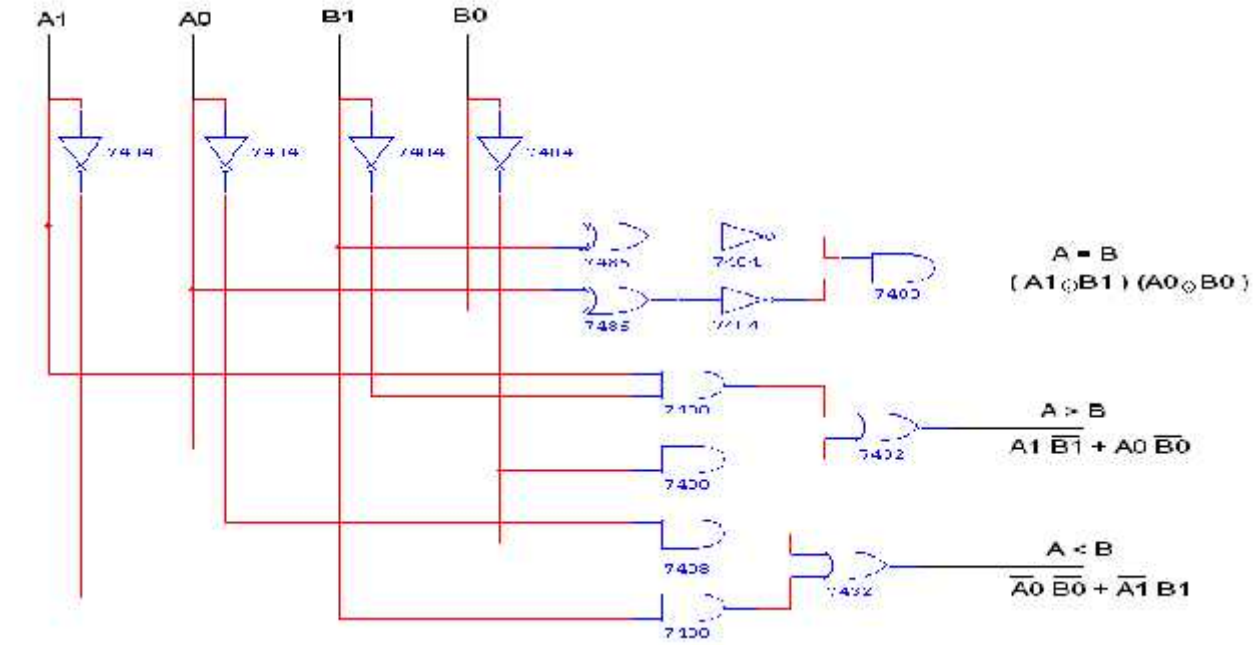
INPUTS		OUTPUTS		
A	B	A > B	A = B	A < B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$A>B = A\bar{B}$

$A<B = \bar{A}B$

$A=B = \bar{A}\bar{B} + AB$

2bit Comparator



$(A>B) = A1\,B1 + A0\,B1\,B0 + B0\,A1\,A0$

$(A=B) = (A0 \oplus B0)\,(A1 \oplus B1)$

$(A<B) = B1\,\bar{A}1 + B0\,\bar{A}1\,\bar{A}0 + \bar{A}0\,B1\,B0$

Truth Table:

INPUTS				OUTPUTS		
A_1	A_0	B_1	B_0	A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

PROCEDURE:

- Check all the components for their working
- Insert the appropriate IC into the IC base
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

RESULT:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment Name: Construction of simple Decoder and Multiplexer circuits using NAND gate.

AIM: To design and implement simple decoder and multiplexer circuits and verify its outputs.

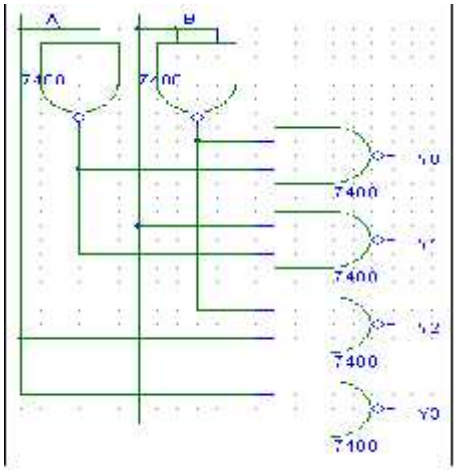
Apparatus Required: Digital trainer kit, patch cords, IC 7400.

Theory:

A decoder is a combinational circuit that connects the binary information from 'n' input lines to a maximum of 2^n unique output lines.
Multiplexers are very useful components in digital systems. They transfer a large number of information units over a smaller number of channels, (usually one channel) under the control of selection signals. Multiplexer means many to one. A multiplexer is a circuit with many inputs but only one output. By using control signals (select lines) we can select any input to the output. Multiplexer is also called as data selector because the output bit depends on the input data bit that is selected. The general multiplexer circuit has 2^n input signals, n control/select signals and 1 output signal.

Logic Diagram:

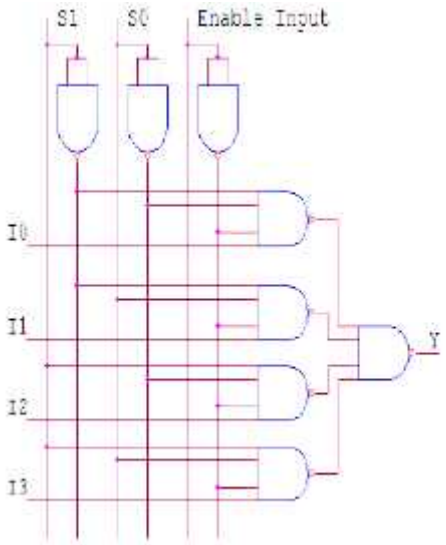
2:4 DECODER:



Truth Table:

INPUT		OUTPUT			
A	B	Y0	Y1	Y2	Y3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

4:1 Multiplexer



Truth Table

Select Inputs		Enable Input	Inputs				Out puts
S_1	S_0	E	I_0	I_1	I_2	I_3	Y
X	X	1	X	X	X	X	0
0	0	0	0	X	X	X	0
0	0	0	1	X	X	X	1
0	1	0	X	0	X	X	0
0	1	0	X	1	X	X	1
1	0	0	X	X	0	X	0
1	0	0	X	X	1	X	1
1	1	0	X	X	X	0	0
1	1	0	X	X	X	1	1

Y = D0 S1' S0' + D1 S1' S0 + D2 S1 S0' + D3 S1 S0

PROCEDURE:

- Check all the components for their working
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

RESULT:

Experiment No.: 05

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment Name: Construction of simple arithmetic circuits-Adder, Subtractor.

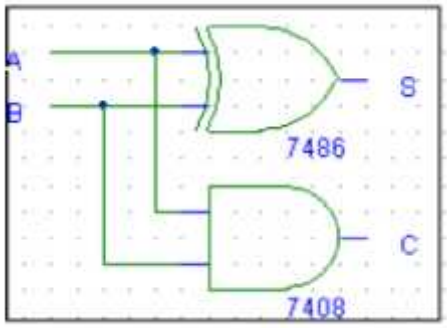
AIM: To design and implement simple adder and subtractor circuits and verify its outputs.

Apparatus Required: Digital trainer kit, patch cords, IC 7400, IC 7408, IC 7486, IC 7432

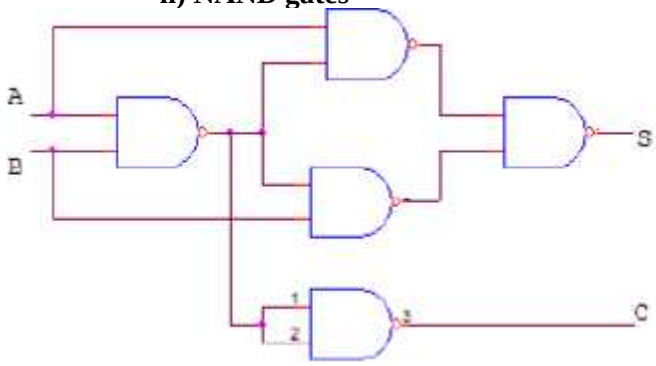
I. To Realize Half Adder

Logic Diagram

i) Basic gates



ii) NAND gates



Truth Table:

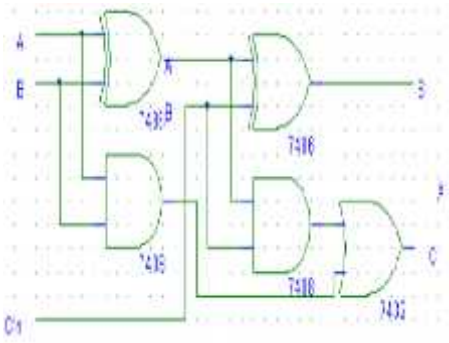
INPUTS		OUTPUTS	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Boolean Expression

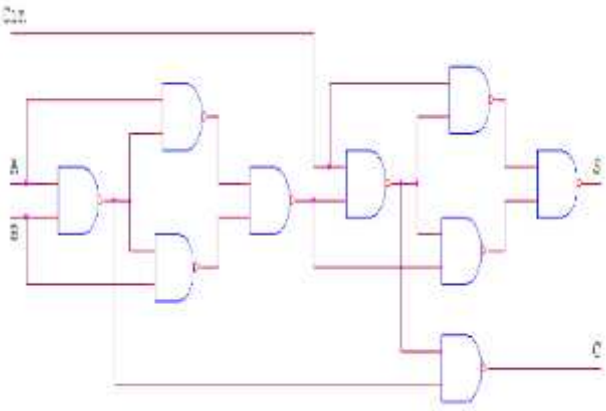
$$S = A \oplus B$$
$$C = A\ B$$

Logic Diagram

i) Basic gates



ii) NAND gates



Truth Table:

TRUTH TABLE

INPUTS			OUTPUTS	
A	B	Cin	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Boolean Expression

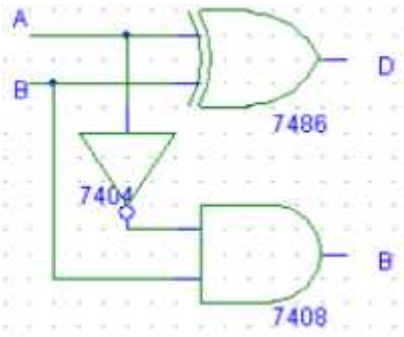
$S = A \oplus B \oplus C$

$C = A\,B + B\,Cin + A\,Cin$

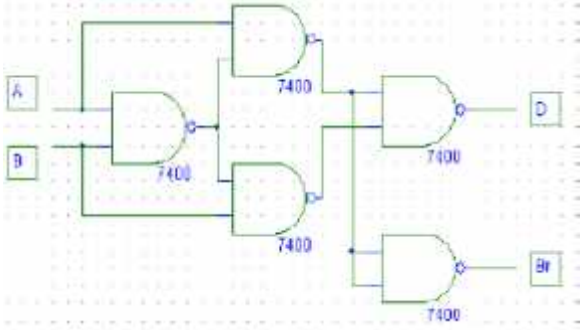
III. Half Subtractor

Logic Diagram

ii) Basic gates



ii) NAND gates



Truth Table:

INPUTS		OUTPUTS	
A	B	D	Br
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Boolean Expression:

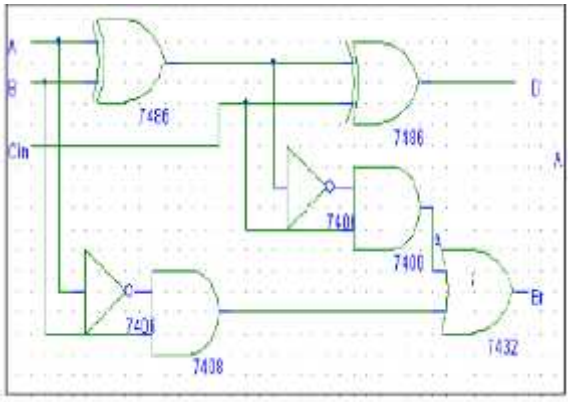
$$D = A \oplus B$$

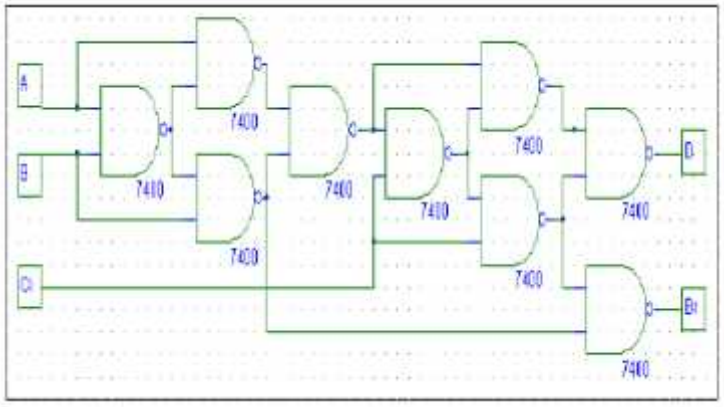
$$Br = \bar{A}B$$

IV. Full Subtractor

Logic Diagram

iii) Basic gates **ii) NAND gates**





Truth Table: **Boolean Expression:**

INPUTS			OUTPUTS	
A	B	Cin	D	Br
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$D = A \oplus B \oplus C$$

$$Br = \bar{A}\,B + B\,Cin + \bar{A}\,Cin$$

PROCEDURE:

- Check all the components for their working
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

RESULT:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No.: 06

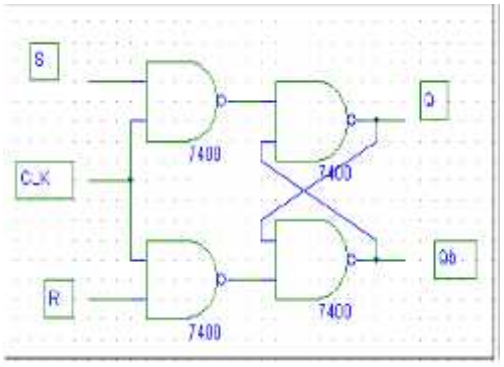
Experiment Name: Realization of RS-JK and D flip-flops using Universal logic gates.

AIM: To design and implement RS-JK and D flip-flops using Universal logic gates and verify its outputs.

Apparatus Required: Digital trainer kit, patch cords, IC 7400, IC 7404, IC 7402, IC 7410, IC 7432

SR Flip Flop

Logic Diagram:

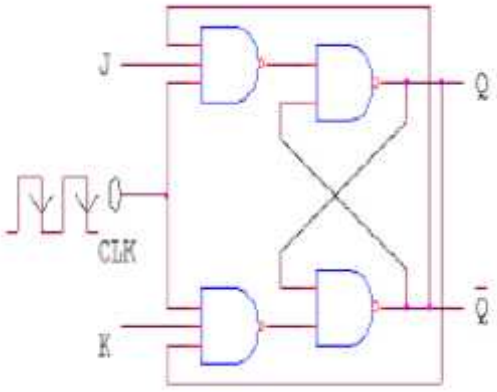


Truth Table

S	R	Q
0	0	No Change
0	1	0
1	0	1
1	1	Forbidden

JK Flip Flop

Logic Diagram



Truth Table

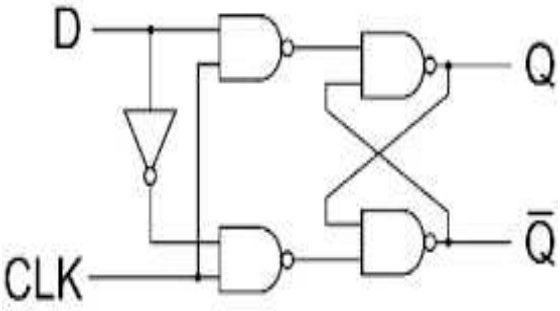
J	K	Q
0	0	No Change
0	1	0
1	0	1
1	1	Race around

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

D Flip Flop

Logic Diagram



Truth Table

D Flip-flop

Symbol

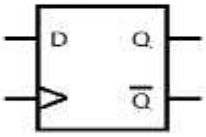


Table of truth:

clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	Q
1	0	0	1
1	1	1	0

PROCEDURE:

- Check all the components for their working
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

RESULT

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No.: 07

Experiment Name: Realization of Ring counter and Johnson's counter.

AIM: To design and implement Ring counter and Johnson's counter and verify its outputs.

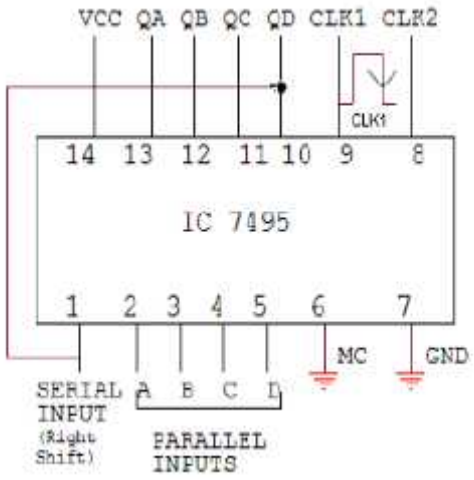
Apparatus Required: Digital trainer kit, patch cords, IC 7404, IC 7495.

Theory:

Ring counter is a basic register with direct feedback such that the contents of the register simply circulate around the register when the clock is running. Here the last output that is QD in a shift register is connected back to the serial input.
A basic ring counter can be slightly modified to produce another type of shift register counter called Johnson counter. Here complement of last output is connected back to the not gate input and not gate output is connected back to serial input. A four bit Johnson counter gives 8 state output.

Ring Counter

Logic Diagram:



Truth Table

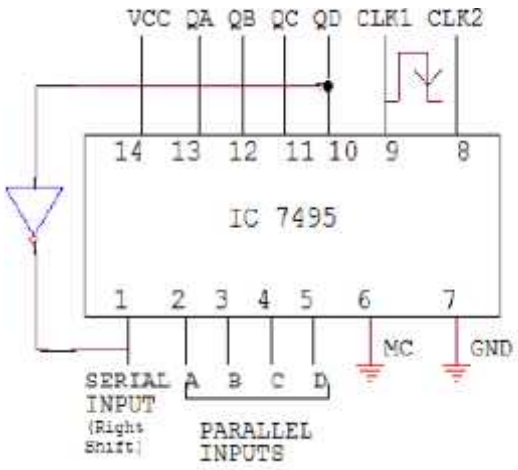
Clock pulses	Q_A	Q_B	Q_C	Q_D
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0
5	0	1	0	0
6	0	0	1	0
7	0	0	0	1
8	1	0	0	0

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Johnson Counter

Logic Diagram:



Truth Table

Clock pulses	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

PROCEDURE:

- Check all the components for their working
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

RESULT

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No.: 08

AIM: Study of Diode as clipper & clamper

APPARATUS REQUIRED:

AC Supply (-12V-0V-12V), PN Diodes-1N4007, Capacitor-470μF,Connecting Wires, Resistor-10 KΩ, Breadboard, Multimeter

THEORY:

There are a variety of diode networks called clippers that have the ability to "clip" off a portion of the input signal without distorting the remaining part of the alternating waveform. The half-wave rectifier is an example of the simplest form of diode clipper—one resistor and diode. Depending on the orientation of the diode, the positive or negative region of the input signal is "clipped" off.
There are two general categories of clippers: series and shunt. The series configuration is defined as one where the diode is in series with the load, while the shunt variety has the diode in a branch parallel to the load.

The clamping network is one that will "clamp" a signal to a different dc level. Thenetwork must have a capacitor, a diode, and a resistive element, but it can also employan independent dc supply to introduce an additional shift. The magnitude of Rand C must be chosen such that the time constant,τ= RC is large enough to ensurethat the voltage across the capacitor does not discharge significantly during the intervalthe diode is non-conducting.

CLIPPER'SCIRCUIT DIAGRAM:

Positive Series Clipper

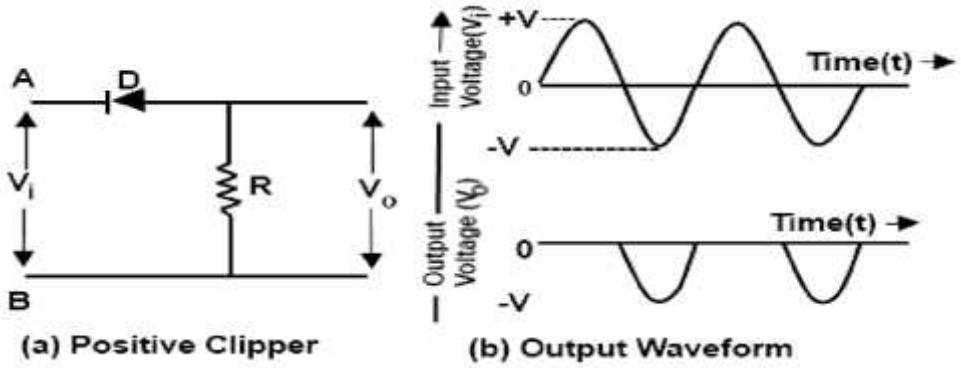


Fig1: Simple positive Series Clipper

Negative Series Clipper

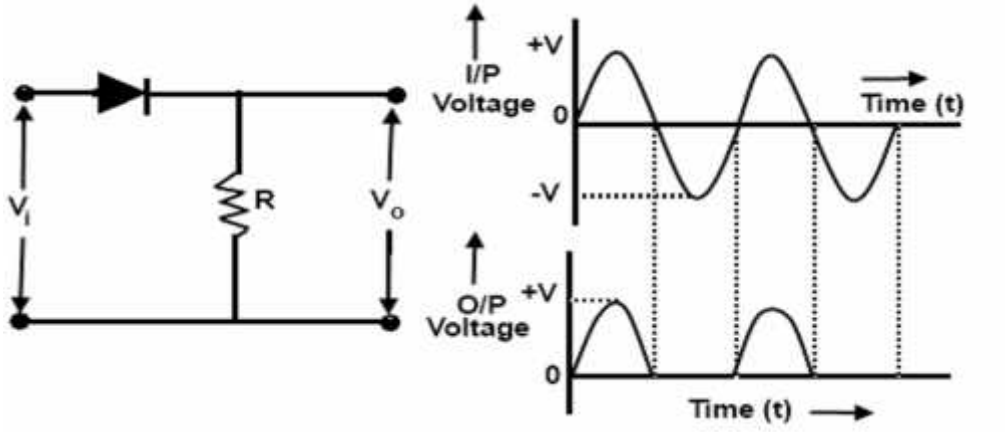


Fig.2: Simple negative series clipper

Biased Positive Series Clipper

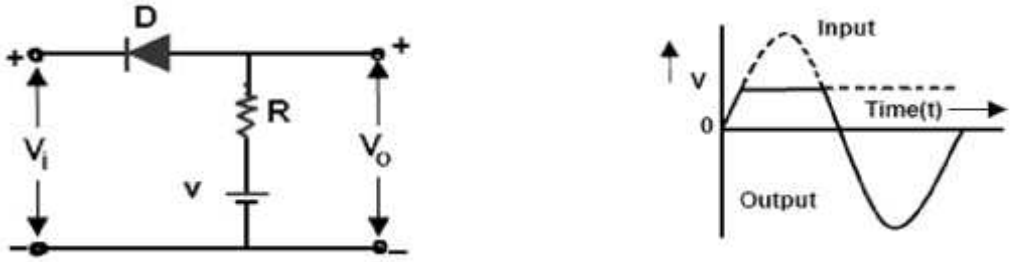


Fig.3: Biased Positive Series Clipper

Biased Negative Series Clipper

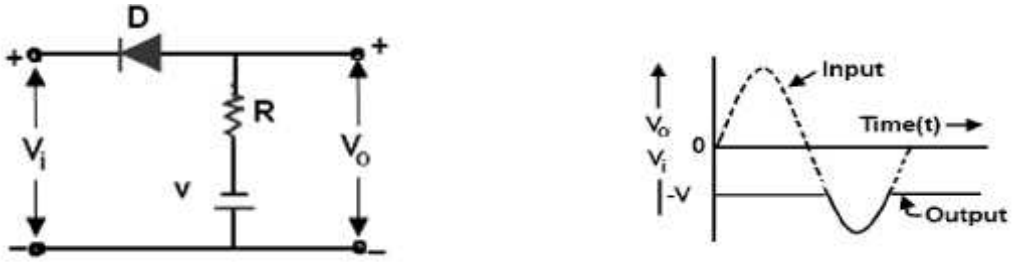


Fig.4: Biased Negative Series Clipper

Positive shunt Clipper

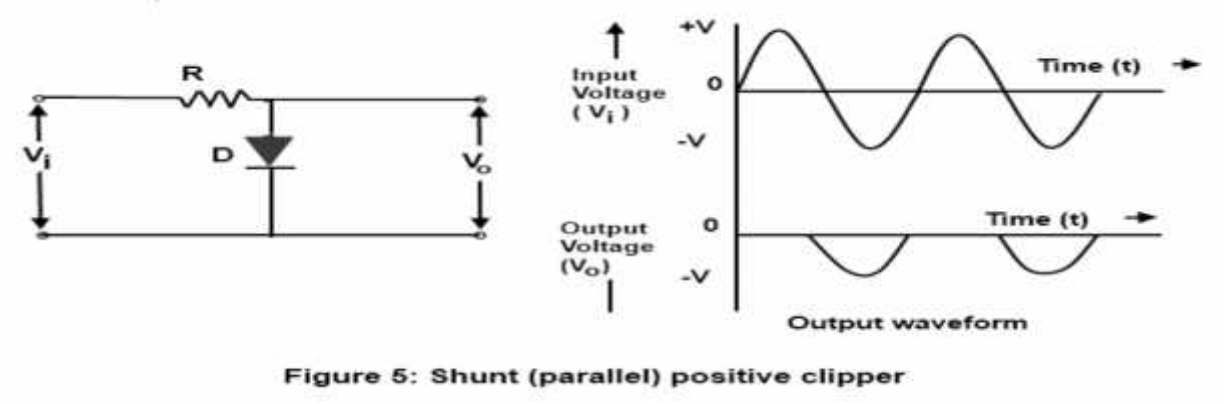


Figure 5: Shunt (parallel) positive clipper

Negative shunt Clipper

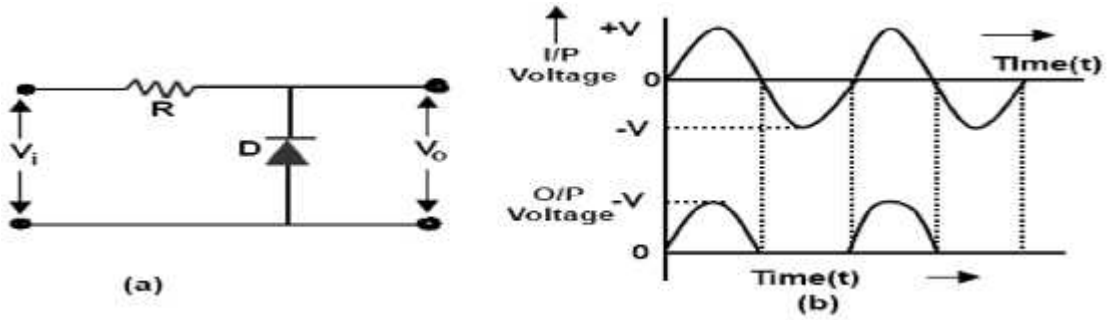


Fig. 6: Shunt (parallel) negative clipper

Biased Positive and negative shunt Clipper

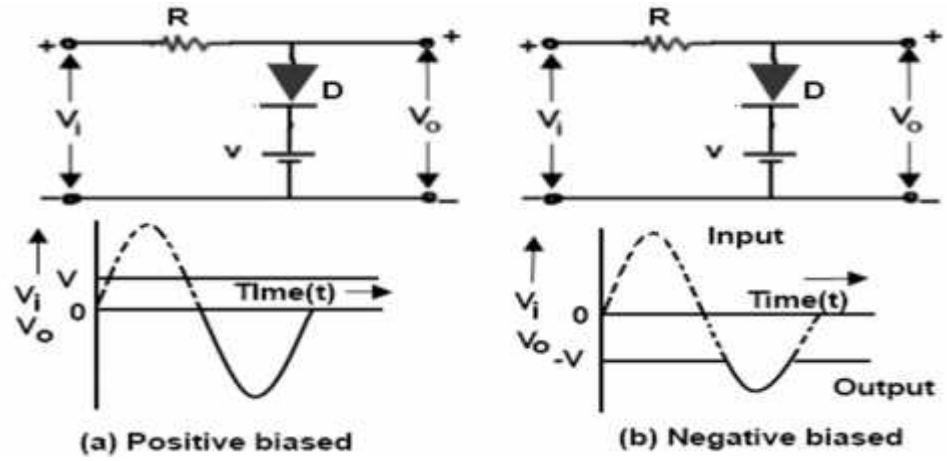
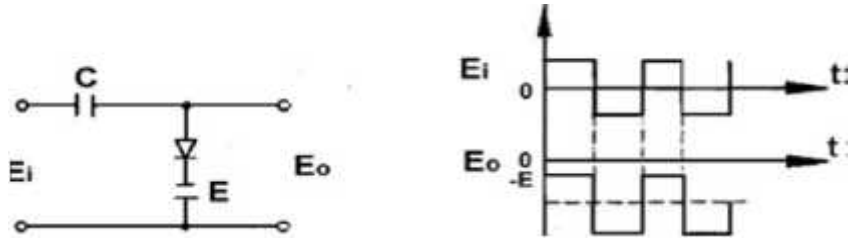


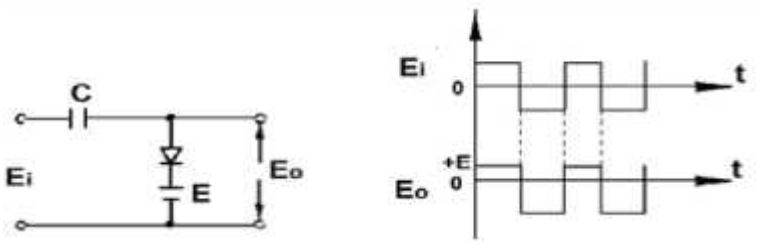
Fig. 7: Biased Positive and negative shunt Clipper

Clamper Circuit Diagrams

Negative Biased Clamping Circuit



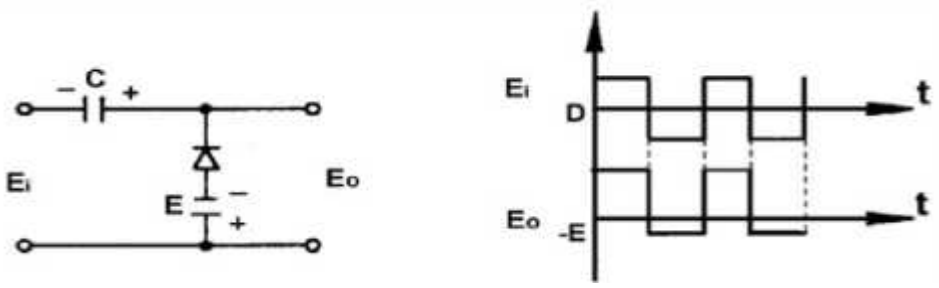
(a)



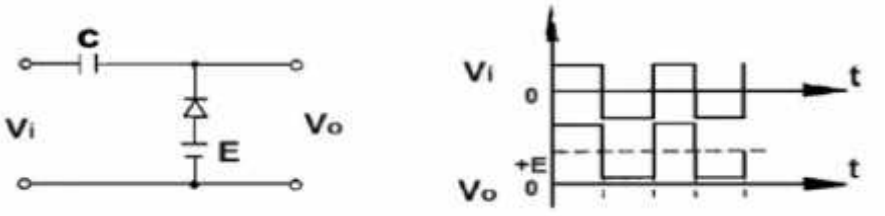
(b)

Fig. 8: Negative Biased Clamping Circuit

Positive Biased Clamping Circuit



(a)



(b)

Fig.9: Positive Biased Clamping Circuit

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No.: 09

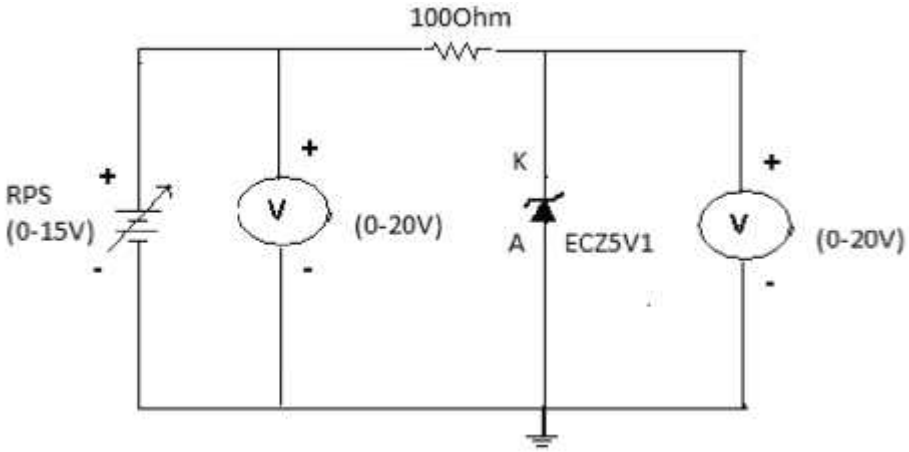
AIM: Study of Zener diode as a voltage regulator

APPARATUS:
Zener diode – ECZ5V1, Regulated Power Supply (0-15V), Voltmeter, Ammeter, Resistor(1KΩ), Breadboard, Connecting wires

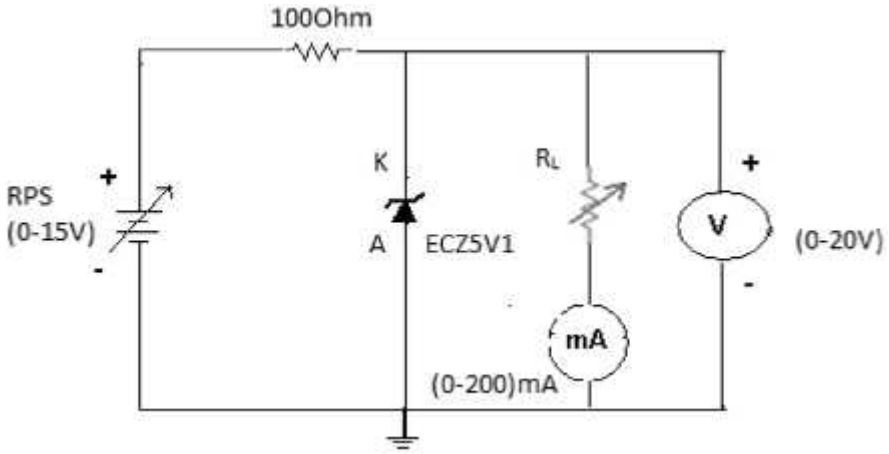
THEORY:
A zener diode is heavily doped p-n junction diode, specially made to operate in thebreak down region. A p-n junction diode normally does not conduct when reverse biased. Butif the reverse bias is increased, at a particular voltage it starts conducting heavily. Thisvoltage is called Break down Voltage. High current through the diode can permanentlydamage the device. To avoid high current, we connect a resistor in series with zener diode.Once the diode starts conducting it maintains almost constant voltage across the terminalswhatever may be the current through it, i.e., it has very low dynamic resistance. It is used involtage regulators.

CIRCUIT DIAGRAM:

SUPPLY SIDE:



LOAD SIDE:



PROCEDURE:

SUPPLY SIDE:
1. Connections are made as per the circuit diagram.
2. The Regulated power supply voltage is increased in steps.
3. For different input voltages (Vi) corresponding output voltages (Vo) are observed and then noted in the tabular form.
4. A graph is plotted between input voltage (Vi) and the output voltage (Vo).

LOAD SIDE:
1. Connection are made as per the circuit diagram
2. The load is placed in full load condition and the output voltage (Vo), load current (IL) are measured.
3. The above step is repeated by decreasing the value of the load in steps.
4. All the readings are tabulated and a graph is plotted between load current (IL) and the output voltage (Vo).

OBSERVATIONS:
SUPPLY SIDE:-

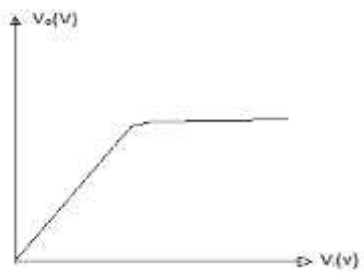
S.NO	Vi (V)	Vo (V)

LOAD SIDE:-

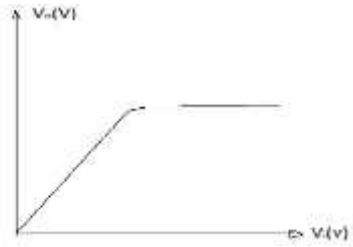
S.NO	IL (V)	Vo(V)

MODEL GRAPH:

SUPPLY SIDE:



LOAD SIDE:



RESULT: Regulator characteristics of zener diode are obtained and graphs are plotted for load and supply side.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Experiment No.: 10

AIM: Study of ripple and regulation characteristics of full wave rectifier without and withcapacitor filter

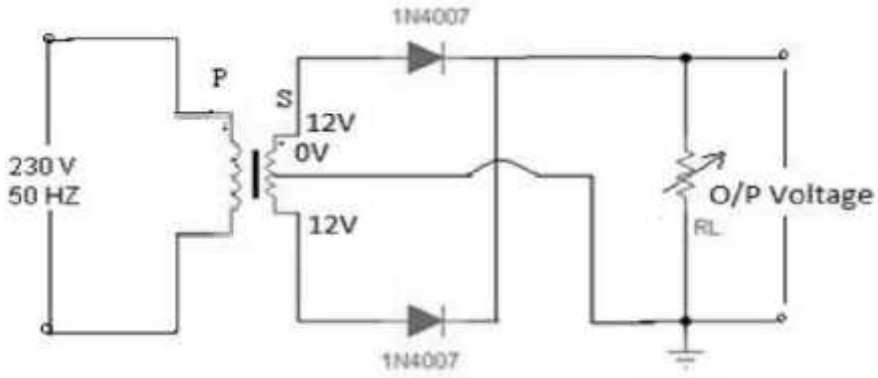
APPARATUS:
AC Supply (12V-0-12V), PN Diodes 1N4007, Capacitor 470μF, Connecting Wires, Variable resistor (0-10) KΩ, Breadboard, Multimeter

THEORY:

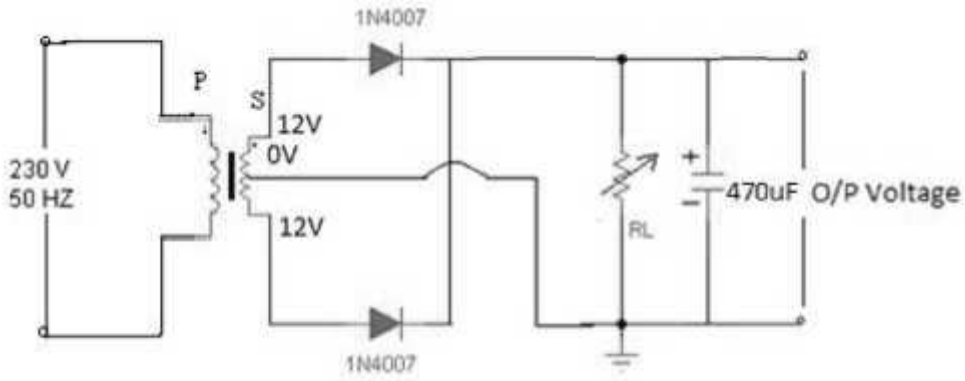
The circuit of a center-tapped full wave rectifier uses two diodes D1&D2. Duringpositive half cycle of secondary voltage (input voltage), the diode D1 is forward biased andD2is reverse biased. The diode D1 conducts and current flows through load resistor RL.During negative half cycle, diode D2 becomes forward biased and D1 reverse biased. Now,D2 conducts and current flows through the load resistor RL in the same direction. There is acontinuous current flow through the load resistor RL, during both the half cycles and will getunidirectional current as show in the model graph. The difference between full wave and halfwave rectification is that a full wave rectifier allows unidirectional (one way) current to theload during the entire 360 degrees of the input signal and half-wave rectifier allows this onlyduring one half cycle (180 degree).

CIRCUIT DIAGRAM:

WITHOUT FILTER



WITH FILTER



PROCEDURE:

1. Connections are made as per the circuit diagram.
2. Connect the ac mains to the primary side of the transformer and the secondary side to the rectifier.
3. Measure the ac voltage at the input side of the rectifier.
4. Measure both ac and dc voltages at the output side the rectifier.
5. Find the theoretical value of the dc voltage by using the formula $Vdc=2Vm/\pi$
6. Connect the filter capacitor across the load resistor and measure the values of Vac and Vdc at the output.
7. The theoretical values of Ripple factors with and without capacitor are calculated.
8. From the values of Vac and Vdc practical values of Ripple factors are calculated. The practical values are compared with theoretical values.

OBSERVATIONS:

WITHOUT FILTER:

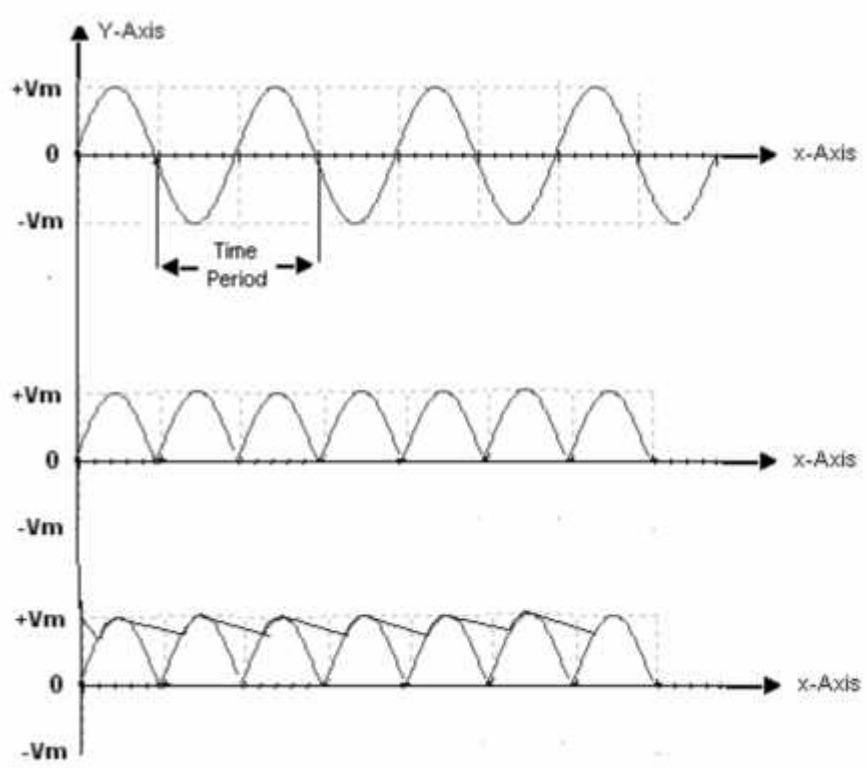
R_L(Ohms)	V_{ac}(Volts)	V_{dc}(Volts)	Ripple Factor = V_{ac}/ V_{dc}	% Regulation (V_{NL}-V_{FL})/V_{FL} *100

WITH FILTER:

RL (Ohms)	Vac (Volts)	Vdc (Volts)	Ripple Factor = Vac/ Vdc	% Regulation (VNL-VFL)/VFL *100

MODEL GRAPHS:

FULLWAVE RECTIFIER (WITH & WITHOUT FILTER):



RESULT:
The ripple factor of the Full-wave rectifier (with filter and without filter) is calculated.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No.: 11

AIM: Study of characteristics curves of B.J.T

APPARATUS:

NPN-Transistor (BC107), Regulated Power Supply (0-15V), Voltmeters (0-20V), Ammeters (0-200μA), (0-200mA), Resistors 1KΩ, Breadboard, Connecting wires, JFET (BFW11)

THEORY:

A transistor is a three terminal device. The terminals are emitter, base, collector. Incommon emitter configuration, input voltage is applied between base and emitter terminalsand output is taken across the collector and emitter terminals. Therefore the emitter terminalis common to both input and output.

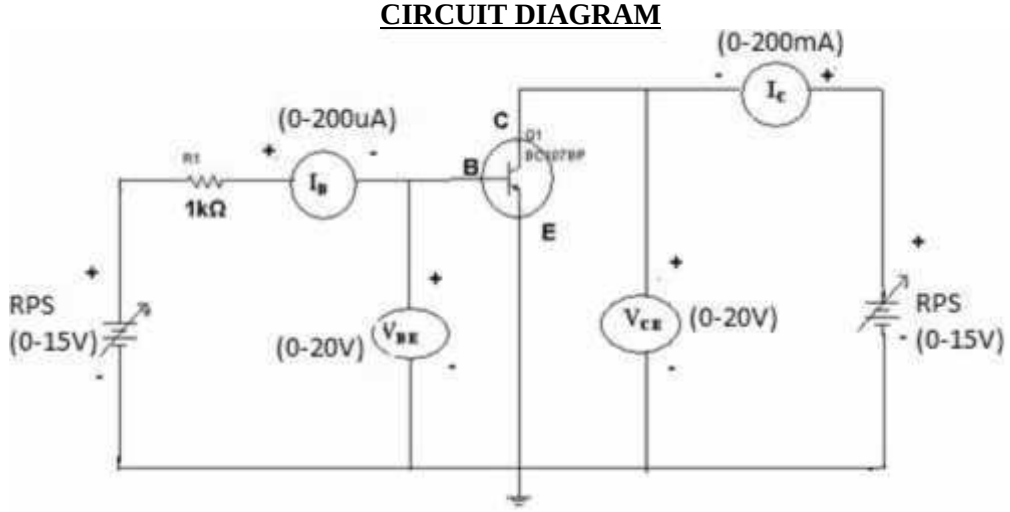
The input characteristics resemble that of a forward biased diode curve. This isexpected since the Base-Emitter junction of the transistor is forward biased. As compared toCB arrangement I_B increases less rapidly with V_{BE}. Therefore input resistance of CE circuitis higher than that of CB circuit.

The output characteristics are drawn between Ic and V_{CE} at constant I_B. the collectorcurrent varies with V_{CE} unto few volts only. After this the collector current becomes almost constant, and independent of V_{CE}. The value of V_{CE} up to which the collector currentchanges with V_{CE} is known as Knee voltage. The transistor always operated in the region above Knee voltage, I_C is always constant and is approximately equal to I_B.

The current amplification factor of CE configuration is given by

$$\beta = \Delta I_C / \Delta I_B$$

CIRCUIT DIAGRAM



PROCEDURE:

INPUT CHARECTERSTICS:

1. Connect the circuit as per the circuit diagram.
2. For plotting the input characteristics the output voltage V_{CE} is kept constant at 1Vand for different values of V_{BE}. Note down the values of I_C
3. Repeat the above step by keeping V_{CE} at 2V and 4V.
4. Tabulate all the readings

OUTPUT CHARACTERSTICS:

1. Connect the circuit as per the circuit diagram.
2. For plotting the output characteristics the input current I_B is kept constant at 10μAand for different values of V_{CE} note down the values of I_C
3. Repeat the above step by keeping I_B at 75 μA, 100 μA
4. Tabulate the all the readings
5. Plot the graph between V_{CE} and I_C for constant I_B.

OBSERVATIONS:

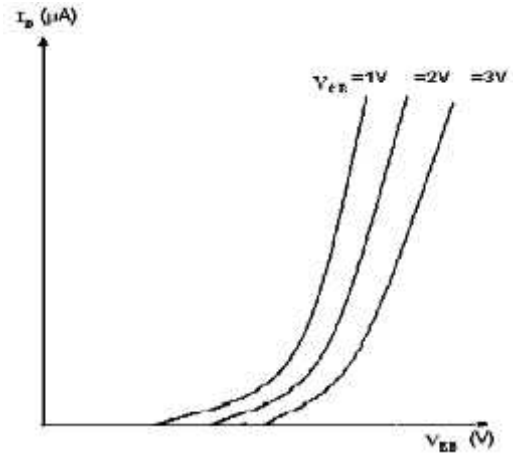
INPUT CHARACTERISTICS:

S.NO	V_{CE} = 1V		V_{CE} = 2V		V_{CE} = 4V	
	V_{BE}(V)	I_B(μA)	V_{BE}(V)	I_B(μA)	V_{BE}(V)	I_B(μA)

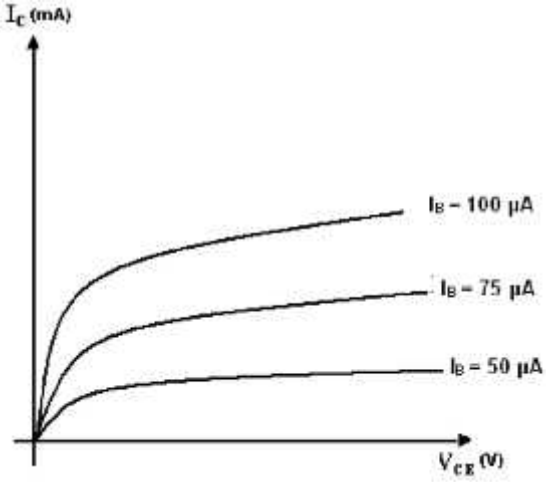
OUTPUT CHARACTERISTICS:

S.NO	IB = 50 μA		IB = 75 μA		IB = 100 μA	
	VCE(V)	IC(mA)	VCE(V)	IC(mA)	VCE(V)	IC(mA)

INPUT CHARACTERSTICS:



OUTPUT CHARECTERSTICS:



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Data structure & Algorithm Lab
Course Code: CS393
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:

1. Develop problem solving ability using Programming.
2. Develop ability to design and analyze algorithms.
3. Introduce students to data abstraction and fundamental data structures.
4. Develop ability to design and evaluate Abstract Data Types and data structures.
5. Apply data structure concepts to various examples and real life applications

Learning Outcomes:

The course will use hands on practice and applying the knowledge gained in theory course to different day to day real world applications..Upon the completion of data structure and algorithm practical course, the student will be able to:

- **Understand** and implement different type of data structure techniques
- **Analyze** the hashing method.
- **Implement** different type of sorting searching techniques.

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: Implementation of array operations
Exercise No. 2: Implementation of linked lists: inserting, deleting a linked list.
Exercise No. 3: Stacks and Queues: adding, deleting elements
Exercise No. 4: Evaluation Problem:Evaluation of infix to postfix expressions on stack.
Exercise No. 5: Circular Queue: Adding & deleting elements
Exercise No. 6: Implementation of stacks using linked lists, Polynomial addition, Polynomial multiplication
Exercise No. 7: Sparse Matrices: Multiplication, addition.
Exercise No. 8: Recursive and Non-recursive traversal of Trees
Exercise No. 9: Threaded binary tree traversal. AVL tree implementation
Exercise No. 10: Application of sorting and searching algorithms

Text Book:

1. Yashavant Kanetkar, Abduln A.P.J. Kalam," Data Structure Through C",2nd edition, BPB Publications
2. Seymour Lipschutz,"Data Structures**",**Revised First edition,McGraw Hill Education.

Recommended Systems/Software Requirements:

1. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
2. Turbo C or TC3 complier in Windows XP or Linux Operating System.

Exercise No.1: Implementation of array operations

Description:

An array is a collection of similar data elements. These data elements have the same data type.The elements of the array are stored in consecutive memory locations and are referenced by an `index`(also known as the subscript). The subscript is an ordinal number which is used to identify an element of the array.There are a number of operations that can be performed on arrays. These operations include:

KATraversing an array
2) Inserting an element in an array
MASearching an element in an array
NADeleting an element from an array

OASorting an array in ascending or descending order

Aim: Write a program to insert a number at a given location in an array.

Algorithm:

The algorithm `INSERT` will be declared as `INSERT(A,N,POS,VAL)`. The arguments are

Step1: `A`, the array in which the element has to be inserted
Step2: `N`, the number of elements in the array
Step3: `POS`, the position at which the element has to be inserted
Step4: `VAL`, the value that has to be inserted

Program:

```
#include <stdio.h>
#include <conio.h>
int main()
{
 int i, n, num, pos, arr[10];
 clrscr();
 printf("\n Enter the number of elements in the array : ");
 scanf("%d", &n);
 for(i=0;i<n;i++)
 {
  printf("\n arr[%d] = ", i);
  scanf("%d", &arr[i]);
 }
 printf("\n Enter the number to be inserted : ");
 scanf("%d", &num);
 printf("\n Enter the position at which the number has to be added : scanf("%d", &pos);
 for(i=n–1;i>=pos;i—)
  arr[i+1] = arr[i];
 arr[pos] = num;
 n = n+1;
 printf("\n The array after insertion of %d is : ", num);
 for(i=0;i<n;i++)
 printf("\n arr[%d] = %d", i, arr[i]);
 getch();
 return 0;
}
```

Input:

```
 Enter the number of elements in the array : 5
 arr[0] = 1
 arr[1] = 2
 arr[2] = 3
 arr[3] = 4
 arr[4] = 5
 Enter the number to be inserted : 0
 Enter the position at which the number has to be added : 3
```

Output:

```
 The array after insertion of 0 is :
 arr[0] = 1
 arr[1] = 2
 arr[2] = 3
 arr[3] = 0
 arr[4] = 4
 arr[5] = 5
```

Aim:Write a program to delete a number from a given location in an array.

Algorithm:

The algorithm `DELETE` will be declared as `DELETE(A,N,POS)`. The arguments are:

Step2: N, the number of elements in the array
Step3: POS, the position from which the element has to be deleted

Program

```
#include <stdio.h>
#include <conio.h>
int main()
{
 int i, n, pos, arr[10];
 clrscr();
 printf("\n Enter the number of elements in the array : ");
 scanf("%d", &n);
 for(i=0;i<n;i++)
 {
  printf("\n arr[%d] = ", i);
  scanf("%d", &arr[i]);
 }
 printf("\nEnter the position from which the number has to be deleted : ");
 scanf("%d", &pos);
 for(i=pos; i<n–1;i++)
  arr[i] = arr[i+1];
 n—;
 printf("\n The array after deletion is : ");
 for(i=0;i<n;i++)
  printf("\n arr[%d] = %d", i, arr[i]);
 getch();
 return 0;
}
```

Input:

Enter the number of elements in the array : 5
arr[0] = 1
arr[1] = 2
arr[2] = 3
arr[3] = 4
arr[4] = 5
Enter the position from which the number has to be deleted : 3

Output:

The array after deletion is :
arr[0] = 1
arr[1] = 2
arr[2] = 3
arr[3] = 5

Lab assignment:

1) Merging two arrays
2) Sorting an array in ascending or descending order

Exercise No. 2: Implementation of linked lists: inserting, deleting a linked list.

Description:

A singly linked list is the simplest type of linked list in which every node contains some data anda pointer to the next node of the same data type. By saying that the node contains a pointer to the next node, we mean that the node stores the address of the next node in sequence.
A new node is added into an already existing linked list like
Case 1: The new node is inserted at the beginning.
Case 2: The new node is inserted at the end.

Case 4: The new node is inserted before a given node.
Before we describe the algorithms to perform insertions in all these four cases, let us first discuss an important term called OVERFLOW. Overflow is a condition that occurs when AVAIL = NULL or no free memory cell is present in the system. When this condition occurs, the program must give an appropriate message.
A node is deleted from an already existing linked list like
Case 1: The first node is deleted.
Case 2: The last node is deleted.
Case 3: The node after a given node is deleted.
Before we describe the algorithms in all these three cases, let us first discuss an important termCalled UNDERFLOW. Underflow is a condition that occurs when we try to delete a node from a linked list that is empty. This happens when START = NULL or when there are no more nodes to delete.
Note that when we delete a node from a linked list, we actually have to free the memory occupied by that node. The memory is returned to the free pool so that it can be used to store other programs and data. Whatever be the case of deletion, we always change the AVAIL pointer so that it points to the address that has been recently vacated.

Algorithm:

Insertion(A) Inserting a Node Before a Given Node in a Linked List

Step 1: IF AVAIL=NULL
Write OVERFLOWGo to Step 12
[END OF IF]
NEW_NODE
Step 2: SET = AVAIL
Step 3: SET AVAIL=AVAIL □□□NEXT
Step 4: SET NEW_NODE ->DATA=VAL
Step 5: SET PTR=START
Step 6: SET PREPTR=PTR
Step 7: Repeat Steps8and9while PTR DATA != NUM
Step 8: SET PREPTR=PTR
Step 9: SET PTR=PTR->NEXT
[END OF LOOP]
Step 10:PREPTR->NEXT = NEW_NODE
Step 11: SET NEW_NODE-> NEXT=PTR
Step 12: EXIT

Insertion(B) Inserting a Node After a Given Node in a Linked List
Step 1: IF AVAIL=NULL
Write OVERFLOW Go to Step 12
[END OF IF]
Step 2: SET = AVAIL->NEW_NODE
Step 3: SET AVAIL=AVAIL->NEXT
Step 4: SET DATA=VAL->NEW_NODE
Step 5: SET PTR=START
Step 6: SET PREPTR=PTR
Step 7: Repeat Steps8and9while PREPTR->DATA!= NUM
Step 8: SET PREPTR=PTR
Step 9: SET PTR=PTR->NEXT
[END OF LOOP]
Step 10: PREPTR->NEXT =NEW_NODE
Step 11: SET NEW_NODE->NEXT=PTR
Step 12: EXIT

Deletion
Step 1: IF START=NULL
Write UNDERFLOW
Go to Step 10
[END OF IF]
Step 2: SET PTR=START

Step 5: SET PREPTR=PTR
Step 6: SET PTR=PTR->NEXT
[END OF LOOP]
Step 7: SET TEMP=PTR
Step 8: SET PREPTR->NEXT=PTR->NEXT
Step 9: FREE TEMP
Step 10:EXIT

Aim:Write a program to create a linked list and perform insertions and deletions Write functions to sort and finally delete the entire list at once.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
struct node
{
 int data;
 struct node *next;
};
struct node *start = NULL;
struct node *create_ll(struct node *);
struct node *display(struct node *);
struct node *insert_beg(struct node *);
struct node *insert_end(struct node *);
struct node *insert_before(struct node *);
struct node *insert_after(struct node *);
struct node *delete_beg(struct node *);
struct node *delete_end(struct node *);
struct node *delete_node(struct node *);
struct node *delete_after(struct node *);
struct node *delete_list(struct node *);
struct node *sort_list(struct node *);
int main(int argc, char *argv[]) {
 int option;
 do
 {
  printf("\n\n *****MAIN MENU *****");
  printf("\n 1:  Create a list");
  printf("\n 2:  Display the list");
  printf("\n 3:  Add a node at the beginning");
  printf("\n 4:  Add a node at the end");
  printf("\n 5:  Add a node before a given node");
  printf("\n 6:  Add a node after a given node");
  printf("\n 7:  Delete a node from the beginning");
printf("\n 8:  Delete a node from the end");
printf("\n 9:  Delete a given node");
printf("\n 10: Delete a node after a given node");
printf("\n 11: Delete the entire list");
printf("\n 12: Sort the list");
printf("\n 13: EXIT");
printf("\n\n Enter your option : ");
  scanf("%d", &option);
  switch(option)
  {
case 1: start = create_ll(start);
 printf("\n LINKED LIST CREATED");
```

```
case 2: start = display(start);
   break;
case 3: start = insert_beg(start);
   break;
case 4: start = insert_end(start);
   break;
case 5: start = insert_before(start);
   break;
case 6: start = insert_after(start);
   break;
case 7: start = delete_beg(start);
   break;
case 8: start = delete_end(start);
   break;
case 9: start = delete_node(start);
   break;
case 10: start = delete_after(start);
   break;
case 11: start = delete_list(start);
 printf("\n LINKED LIST DELETED");
   break;
case 12: start = sort_list(start);
   break;
  }
}while(option !=13);
return 0;
struct node *create_ll(struct node *start)
struct node *new_node, *ptr;
printf("\n Enter -1 to end");
printf("\n Enter the data : ");
scanf("%d", &num);
while(num!=-1)
new_node = (struct node*)malloc(sizeof(struct node));
new_node -> data=num;
  if(start==NULL)
  {
 new_node -> next = NULL;
   start =
new_node;
  }
  else
  {
   ptr=start;
   while(ptr->next!=NULL)
   ptr=ptr->next;
   ptr->next =
new_node;
   new_node->next=NULL;
  }
  printf("\n Enter the data : ");
  scanf("%d", &num);
 }
 return start;
}
struct node *display(struct node *start)
{
```

```
 ptr = start;
 while(ptr != NULL)
 {
  printf("\t %d", ptr -> data);
  ptr = ptr -> next;
 }
 return start;
}
struct node *insert_beg(struct node *start)
{
 struct node *new_node;
 int num;
 printf("\n Enter the data : ");
 scanf("%d", &num);
 new_node = (struct node *)malloc(sizeof(struct node));
 new_node -> data = num;
 new_node -> next = start;
 start = new_node;
 return start;
}
struct node *insert_end(struct node *start)
{
 struct node *ptr, *new_node;
 int num;
 printf("\n Enter the data : ");
 scanf("%d", &num);
 new_node = (struct node *)malloc(sizeof(struct node));
 new_node -> data = num;
 new_node -> next = NULL;
 ptr = start;
 while(ptr -> next != NULL)
 ptr = ptr -> next;
 ptr -> next = new_node;
 return start;
}
struct node *insert_before(struct node *start)
{
 struct node *new_node, *ptr, *preptr;
 int num, val;
 printf("\n Enter the data : ");
 scanf("%d", &num);
 printf("\n Enter the value before which the data has to be inserted : ");
 scanf("%d", &val);
 new_node = (struct node *)malloc(sizeof(struct node));
 new_node -> data = num;
 ptr = start;
 while(ptr -> data != val)
 {
  preptr = ptr;
  ptr = ptr -> next;
 }
 preptr -> next = new_node;
 new_node -> next = ptr;
 return start;
}
struct node *insert_after(struct node *start)
```

```
 struct node *new_node, *ptr, *preptr;
 int num, val;
 printf("\n Enter the data : ");
 scanf("%d", &num);
 printf("\n Enter the value after which the data has to be inserted : ");
scanf("%d", &val);
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> data = num;
ptr = start;
preptr = ptr;
while(preptr -> data != val)
 {
 preptr = ptr;
 ptr = ptr -> next;
 }
preptr -> next=new_node;
new_node -> next = ptr;
return start;
struct node *delete_beg(struct node *start)
struct node *ptr;
ptr = start;
start = start -> next;
 free(ptr);
return start;
struct node *delete_end(struct node *start)
struct node *ptr, *preptr;
ptr = start;
while(ptr -> next != NULL)
 {
 preptr = ptr;
 ptr = ptr -> next;
 }
preptr -> next = NULL;
 free(ptr);
return start;
struct node *delete_node(struct node *start)
struct node *ptr, *preptr;
int val;
printf("\n Enter the value of the node which has to be deleted : ");
scanf("%d", &val);
ptr = start;
if(ptr -> data == val)
 {
start = delete_beg(start);
return start;
}
 else
 {
while(ptr -> data != val)
  {
   preptr = ptr;
   ptr = ptr -> next;
  }
  preptr -> next = ptr -> next;
  free(ptr);
  return start;
```

```
}
struct node *delete_after(struct node *start)
{
 struct node *ptr, *preptr;
 int val;
 printf("\n Enter the value after which the node has to deleted : ");
 scanf("%d", &val);
 ptr = start;
 preptr = ptr;
 while(preptr -> data != val)
 {
  preptr = ptr;
  ptr = ptr -> next;
 }
 preptr -> next=ptr -> next;
 free(ptr);
 return start;
}
struct node *delete_list(struct node *start)
{
            struct
            node     *ptr;
 if(start!=NULL){
  ptr=start;
  while(ptr != NULL)
  {
   printf("\n %d is to be deleted next", ptr -> data);
   start =
delete_beg(ptr);
   ptr =
start;
  }
 }

 return start;
}
struct node *sort_list(struct node *start)
{
 struct node *ptr1, *ptr2;
 int temp;
 ptr1 = start;
 while(ptr1 -> next != NULL)
 {
  ptr2 = ptr1 -> next;
  while(ptr2 != NULL)
  {
   if(ptr1 -> data > ptr2 -> data)
   {
            temp = ptr1 -> data;
            ptr1 -> data = ptr2 -> data;
            ptr2 -> data = temp;
   }
   ptr2 = ptr2 -> next;
  }
  ptr1 = ptr1 -> next;
 }
```

```
}
```

Input:

```
3
4
5
```

Output:

```
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add the node at the end
5: Add the node before a given node
6: Add the node after a given node
7: Delete a node from the beginning
8: Delete a node from the end
9: Delete a given node
10: Delete a node after a given node
11: Delete the entire list
12: Sort the list
13: Exit

Enter your option : 1
Enter the data :3
Enter your option : 2
3
Enter your option : 3
Enter the data : 4
Enter your option : 6
Add after given node:4
Enter the data : 5
Enter your option : 2
4       5       3
Enter your option : 10
Delete after a given node:5
Enter your option : 2
4       5
```

Lab Assignment:

1) WAP to implement circular linked list.
2) WAP to insert and delete an element in a doubly linked list(all cases).

Exercise No. 3: Stacks and Queues: adding, deleting elements

Description:

A stack is a linear data structure which uses the same principle, i.e., the elements in a stack are added and removed only from one end, which is called the`TOP`. Hence, a stack is called a LIFO (Last-In First-Out) datastructure, as the element that was inserted last is the first one to be taken out.

A stack supports three basic operations: `push`, `pop`, and `peek`. The `push` operation adds an element to the top of the stack and the `pop` operation removes the element from the top of the stack. The `peek` operation returns the value of the topmost element of the stack.

Aim: Write a program to perform Push, Pop, and Peek operations on a stack.

Algorithm:

Insertion:

```
Step 1: IF TOP=MAX-1
                PRINT OVERFLOW
        Goto Step 4
```

Step 2: SET TOP=TOP+1
Step 3: SET STACK[TOP]=VALUE
Step 4: END

Deletion:
Step 1: IF TOP=NULL
PRINT UNDERFLOW
Goto Step 4
[END OF IF]
Step 2: SET VAL=STACK[TOP]
Step 3: SET TOP=TOP-1
Step 4: END

Peek:
Step 1: IF TOP=NULL
PRINT STACK IS EMPTY
Goto Step 3
Step 2: RETURN STACK[TOP]
Step 3: END

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 3 // Altering this value changes size of stack created
int st[MAX], top=-1;
void push(int st[], int val);
int pop(int st[]);
int peek(int st[]);
void display(int st[]);
int main(int argc, char *argv[]) {
 int val, option;
 do
 {
  printf("\n *****MAIN MENU*****");
  printf("\n 1. PUSH");
  printf("\n 2. POP");
  printf("\n 3. PEEK");
  printf("\n 4. DISPLAY");
  printf("\n 5. EXIT");
  printf("\n Enter your option: ");
  scanf("%d", &option);
  switch(option)
  {
  case 1:
   printf("\n Enter the number to be pushed on stack: ");
   scanf("%d", &val);
   push(st, val);
   break;
  case 2:
   val = pop(st);
   if(val != -1)
   printf("\n The value deleted from stack is: %d", val);
   break;
  case 3:
```

```
   if(val != -1)
   printf("\n The value stored at top of stack is: %d", val);
   break;
  case 4:
   display(st);
   break;
  }
 }while(option != 5);
 return 0;
}
void push(int st[], int val)
{
 if(top == MAX-1)
 {
  printf("\n STACK OVERFLOW");
 }
 else
 {
  top++;
  st[top] = val;
 }
}
int pop(int st[])
{
 int val;
 if(top == -1)
 {
  printf("\n STACK UNDERFLOW");
  return -1;
 }
 else
 {
  val = st[top];
  top--;
  return val;
 }
}
void display(int st[])
{
 int i;
 if(top == -1)
 printf("\n STACK IS EMPTY");
 else
 {
  for(i=top;i>=0;i--)
  printf("\n %d",st[i]);
  printf("\n"); // Added for formatting purposes
 }
}
int peek(int st[])
{
 if(top == -1)
 {
  printf("\n STACK IS EMPTY");
  return -1;
 }
```

```
 return (st[top]);
}
```

Output

```
*****MAIN MENU*****
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter your option : 1
Enter the number to be pushed on stack : 500
Enter your option : 1
Enter the number to be pushed on stack : 700
Enter your option : 4
700     500
Enter your option : 3
Enter your option : 4
700
Enter your option : 2
Enter your option : 4
500
```

Description:

A queue is a FIFO (First-In, First-Out) data structure in which the element that is inserted first is the first one to be taken out. The elements in a queue are added at one end called the REAR and removed from the other end called the FRONT. Queues can be implemented by using either arrays or linked lists.

Aim: Write a program to perform Insertion, Deletion, and Peek operations on a queue.

Algorithm:

Insertion:

```
Step 1: IF REAR=MAX-1
                Write OVERFLOW
        Goto step 4
        [END OF IF]
Step 2: IF FRONT=-1 and REAR=-1
                SET FRONT=REAR =ELSE
                SET REAR=REAR+1
        [END OF IF]
Step 3: SET QUEUE[REAR]=NUM
Step 4: EXIT
```

Deletion:

```
Step 1: IF FRONT=-1OR FRONT>REAR
                Write UNDERFLOW
        ELSE
                SET VAL=QUEUE[FRONT]
                SET FRONT=FRONT+1
        [END OF IF]
Step 2: EXIT
```

Program:

```
#include <stdio.h>
#include <conio.h>
#define MAX 10 // Changing this value will change length of array
int queue[MaX];
int front = -1, rear = -1;
void insert(void);
int delete_element(void);
int peek(void);
void display(void);
int main()
```

```
 do
 {
  printf("\n\n ***** MAIN MENU *****");
  printf("\n 1. Insert an element");
  printf("\n 2. Delete an element");
  printf("\n 3. Peek");
  printf("\n 4. Display the queue");
  printf("\n 5. EXIT");
  printf("\n Enter your option : ");
  scanf("%d", &option);
  switch(option)
  {
  case 1:
   insert();
   break;
  case 2:
   val = delete_element();
   if (val != -1)
   printf("\n The number deleted is : %d", val);
   break;
  case 3:
   val = peek();
   if (val != -1)
   printf("\n The first value in queue is : %d", val);
   break;
  case 4:
   display();
   break;
  }
 }while(option != 5);
 getch();
 return 0;
}
void insert()
{
 int num;
 printf("\n Enter the number to be inserted in the queue : ");
 scanf("%d", &num);
 if(rear == MAX-1)
 printf("\n OVERFLOW");
 else if(front == -1 && rear == -1)
 front = rear = 0;
 else
 rear++;
 queue[rear] = num;
}
int delete_element()
{
 int val;
if(front == -1 || front>rear)
 {
 printf("\n UNDERFLOW");
 return -1;
 }
 else
 {
 val = queue[front];
 front++;
 if(front > rear)
 front = rear = -1;
 return val;
```

```
if(front==-1 || front>rear)
 {
 printf("\n QUEUE IS EMPTY");
 return -1;
 }
 else
 {
 return queue[front];
 }
void display()
int i;
printf("\n");
if(front == -1 || front > rear)
printf("\n QUEUE IS EMPTY");
 else
 {
 for(i = front;i <= rear;i++)
 printf("\t %d", queue[i]);
 }
```

Output:

```
***** MAIN MENU *****"
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option : 1
Enter the number to be inserted in the queue : 50
```

Exercise No. 4: Evaluation Problem: Evaluation of infix to postfix expressions on stack.

Description:

Infix, postfix, and prefix notations are three different but equivalent notations of writing algebraic expressions. For example, if an expression is written as `A+B` in infix notation, the same expression can be written as `AB+` in postfix notation. The order of evaluation of a postfix expression is always from left to right. Even brackets cannot alter the order of evaluation. The expression `(A+B)*C` can be written as: `[AB+]*C` =>`AB+C*` in the postfix notation.

Aim:Write a program to convert a given infix expression into its postfix Equivalent, Implement the stack using an array.

Algorithm:

Step 1: Add)to the end of the infix expression
Step 2: Push(onto the stack
Step 3: Repeat until each character in the infix notation is scanned
IF a(is encountered, push it on the stack
IF an operand (whetheradigit oracharacter) is encountered, add it to thepostfix expression.
IF a)is encountered, then
    a. Repeatedly pop from stack and add it to the postfix expression until a
(is encountered.
    b. Discard the (.That is, remove the(from stack and do notadd it to the postfix expression
IF an operator is encountered, then
    a. Repeatedly pop from stack and add each operator (popped from the stack) to thepostfix expression
which has the same precedence orahigher precedence than)
    b. Push the operator to the stack
[END OF IF]
Step 4: Repeatedly pop from the stack and add it to the postfix expression until the stack is empty
Step 5: EXIT

Program:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
```

```
char stack[MAX];
int top=1;
char pop();  /*declaration of  pop function*/
void push(char item); /*declaration of  push function*/
int prcd(char symbol)  /*checking the precedence*/
{
   switch(symbol) /*assigning values for symbols*/
   {
      case '+':
      case '-': return 2;
      break;
      case '*':
      case '/': return 4;
      break;
      case '^':return 6;
      break;
      case '(':
      case ')':
      case '#':return 1;
     break;
  }
}
int(isoperator(char symbol))  /*assigning  operators*/
{
   switch(symbol)
   {
       case '+':
       case '*':
   case '-':
      case '/':
      case '^':
      case '(':
      case ')':return 1;
      break;
      default:return 0;
   }
 }
/*converting infix to postfix*/
void convertip(char infix[],char postfix[])
{
 int i,symbol,j=0;
 stack[++top]='#';
 for(i=0;i<strlen(infix);i++)
 {
  symbol=infix[i];
  if(isoperator(symbol)==0)
  {
     postfix[j]=symbol;
      j++;
  }
  else
  {
     if(symbol=='(')
     push(symbol);   /*function call  for pushing elements into the stack*/
     else if(symbol==')')
     {
```

```
      {
         postfix[j]=pop();
         j++;
      }
      pop();     /*function call  for popping elements into the stack*/
   }
  else
  {
      if(prcd(symbol)>prcd(stack[top]))
      push(symbol);
      else
      {
        while(prcd(symbol)<=prcd(stack[top]))
        {
          postfix[j]=pop();
           j++;
        }
       push(symbol);
      }/*end of else loop*/
    }/*end of else loop*/
   } /*end of else loop*/
  }/*end of for loop*/
 While (stack[top]!='#')
 {
    postfix[j]=pop();
     j++;
  }
  postfix[j]='\0'; /*null terminate string*/
}
/*main program*/
void main()
 {
     char infix[20],postfix[20];
     printf("enter the valid infix string \n");
     gets(infix);
     convertip(infix,postfix);    /*function call   for converting infix to postfix */
     printf("the corresponding postfix string is:\n");
    puts(postfix);
 }
/*push operation*/
void push(char item)
{
   top++;
   stack[top]=item;
 }
/*pop operation*/
char pop()
{
   char a;
   a=stack[top];
   top--;
   return a;
 }
```

Input:

A+B*C

Output:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Exercise No. 5: Circular Queue: Adding & deleting elements

Description:

In the circular queue, the first index comes right after the last index.The circular queue will be full only when `FRONT=0` and `REAR=Max-1`. A circular queue is implemented in the same manner as a linear queue is implemented.

Aim: Write a program to implement a circular queue using array.

Algorithm:

Insertion:

```
Step 1: IF FRONT = and Rear=MAX-1
                Write OVERFLOW
                Goto step 4
        [End OF IF]

Step 2:
IF FRONT=-1 and REAR=-1
        SET FRONT=REAR =0
ELSE IF REAR=MAX-1and FRONT !=0
                SET REAR =0
   ELSE
                SET REAR=REAR+1
[END OF IF]
Step 3: SET QUEUE[REAR]=VAL
Step 4: EXIT
Deletion:
Step 1: IF FRONT=-1
                Write UNDERFLOW
                Goto Step 4
        [END of IF]
Step 2: SET VAL=QUEUE[FRONT]
Step 3: IF FRONT=REAR
                SET FRONT=REAR=-1
        ELSE
                IF FRONT=MAX -1
                        SET FRONT =0
                ELSE
                        SET FRONT=FRONT+1
                [END of IF]
        [END OF IF]
Step 4: EXIT
```

Program:

```
#include <stdio.h>
#include <conio.h>
#define MAX 10
int queue[MAX];
int front=–1, rear=–1;
void insert(void);
int delete_element(void);
int peek(void);
void display(void);
int main()
{
 int option, val;
 clrscr();
 do
 {
  printf("\n ***** MAIN MENU *****");
  printf("\n 1. Insert an element");
  printf("\n 2. Delete an element");
  printf("\n 3. Peek");
```

```
 printf("\n Enter your option : ");
 scanf("%d", &option);
 switch(option)
 {
  case 1:
   insert();
   break;
  case 2:
   val = delete_element();
   if(val!=–1)
   printf("\n The number deleted is : %d", val);
   break;
  case 3:
   val = peek();
   if(val!=–1)
   printf("\n The first value in queue is : %d",     break;
  case 4:
   display();
   break;
 }
}while(option!=5);
getch();
return 0;
}
void insert()
{
 int num;
 printf("\n Enter the number to be inserted in the queue : ");
 scanf("%d", &num);
 if(front==0 && rear==MAX–1)
  printf("\n OVERFLOW");
 else if(front==–1 && rear==–1)
 {
  front=rear=0;
  queue[rear]=num;
 }
 else if(rear==MAX–1 && front!=0)
 {
  rear=0;
  queue[rear]=num;
 }
 else
 {
  rear++;
  queue[rear]=num;
 }
}
int delete_element()
{
 int val;
 if(front==–1 && rear==–1)
  {
  printf("\n UNDERFLOW");
  return –1;
  }
 val = queue[front];
 if(front==rear)
  front=rear=–1;
 else
 {
  if(front==MAX–1)
```

```
  front++;
 }
 return val;
}
int peek()
{
 if(front==–1 && rear==–1)
 {
  printf("\n QUEUE IS EMPTY");
  return –1;
}
 else
 {
  return queue[front];
 }
}
void display()
{
 int i;
 printf("\n");
 if (front ==–1 && rear= =–1)
  printf ("\n QUEUE IS EMPTY");
 else
 {
  if(front<rear)
  {
   for(i=front;i<=rear;i++)
    printf("\t %d", queue[i]);
  }
  else
  {
   for(i=front;i<MAX;i++)
    printf("\t %d", queue[i]);
   for(i=0;i<=rear;i++)
    printf("\t %d", queue[i]);
  }
 }
}
```

Output

```
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option : 1
Enter the number to be inserted in the queue : 25
Enter your option : 2
The number deleted is : 25
Enter your option : 3
QUEUE IS EMPTY
Enter your option : 5
```

Exercise No. 6: Implementation of Polynomial addition, Polynomial multiplicationusing linked lists.

Description:

A polynomial is represented in the memory using a linked list. Consider a polynomial $6x^3+9x^2+7x+1$. Every individual term in a polynomial consists of two parts, a coefficientand a power. Here, 6, 9, 7, and 1 are the coefficients of the terms that have 3, 2, 1, and 0 as theirpowers respectively.

Every term of a polynomial can be represented as a node of the linked list

Aim:Write a program to add two polynomials.

Program:

```
#include <stdio.h>
typedef struct pnode
{
float coef;
int exp;
struct pnode *next;
}p;
p *getnode();
void main()
{
p *p1,*p2,*p3;

p *getpoly(),*add(p*,p*);

void display(p*);
clrscr();
printf("\n enter first polynomial");
p1=getpoly();
printf("\n enter second polynomial");
p2=getpoly();
printf("\nthe first polynomial is");
display(p1);
printf("\nthe second polynomial is");
display(p2);
p3=add(p1,p2);
printf("\naddition of two polynomial is :\n");
display(p3);

}
p *getpoly()
{
p *temp,*New,*last;
int flag,exp;
char ans;
float coef;
temp=NULL;
flag=1;
printf("\nenter the polynomial in descending order of exponent");
do
{
printf("\nenter the coef & exponent of a term");
scanf("%f%d",&coef,&exp);
New=getnode();
if(New==NULL)
printf("\nmemory cannot be allocated");
New->coef=coef;
New->exp=exp;
if(flag==1)
{
temp=New;
last=temp;
flag=0;
}
else
{
last->next=New;
last=New;
}
```

```
ans=getch();
}
while(ans=='y');
return(temp);
}
p *getnode()
{
p *temp;
temp=(p*) malloc (sizeof(p));
temp->next=NULL;
return(temp);
}
void display(p*head)
{
p*temp;
temp=head;
if(temp==NULL)
printf("\npolynomial empty");
while(temp->next!=NULL)
{
printf("%0.1fx^%d+",temp->coef,temp->exp);
temp=temp->next;
}
printf("\n%0.1fx^%d",temp->coef,temp->exp);
getch();
}
p*add(p*first,p*second)
{
p *p1,*p2,*temp,*dummy;
char ch;
float coef;
p *append(int,float,p*);
p1=first;
p2=second;
temp=(p*)malloc(sizeof(p));
if(temp==NULL)
printf("\nmemory cannot be allocated");
dummy=temp;
while(p1!=NULL&&p2!=NULL)
{
if(p1->exp==p2->exp)
{
coef=p1->coef+p2->coef;
temp=append(p1->exp,coef,temp);
p1=p1->next;
p2=p2->next;
}
else
if(p1->expexp)
{
coef=p2->coef;
temp=append(p2->exp,coef,temp);
p2=p2->next;
}
else
if(p1->exp>p2->exp)
{
coef=p1->coef;
temp=append(p1->exp,coef,temp);
p1=p1->next;
}
```

```
{
temp=append(p1->exp,p1->coef,temp);
p1=p1->next;
}
while(p2!=NULL)
{
temp=append(p2->exp,p2->coef,temp);
p2=p2->next;
}
temp->next=NULL;
temp=dummy->next;
free(dummy);
return(temp);
}
p*append(int Exp,float Coef,p*temp)
{
p*New,*dum;
New=(p*)malloc(sizeof(p));
if(New==NULL)
printf("\ncannot be allocated");
New->exp=Exp;
New->coef=Coef;
New->next=NULL;
dum=temp;
dum->next=New;
dum=New;
return(dum);
}
```

Input:

A^2+2A+2

A^3+3A+3

Output:

A^3+A^2+5A+5

Lab Assignment:

1) Write a program to multiply two polynomials.

Exercise No. 7: Sparse Matrices: Multiplication, addition.

Description:

Sparse matrix is a matrix that has large number of elements with a zero value. In order to efficiently utilize the memory, specialized algorithms and data structures that take advantage of the sparse structure should be used. If we apply the operations using standard matrix structures and algorithms to sparse matrices, then the execution will slow down and the matrix will consume large amount of memory. Sparse data can be easily compressed, which in turn can significantly reduce memory usage.

Aim: Write a program to multiply sparse matrices.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#define MAX1 3
#define MAX2 3
#define MAXSIZE 20
#define TRUE 1
#define FALSE 2
struct sparse
{
int *sp ;
int row ;
```

```
} ;
void initsparse ( struct sparse * ) ;
void create_array ( struct sparse * ) ;
int count ( struct sparse ) ;
void display ( struct sparse ) ;
void create_tuple ( struct sparse*, struct sparse ) ;
void display_tuple ( struct sparse ) ;
void prodmat ( struct sparse *, struct sparse, struct sparse ) ;
void searchina ( int *sp, int ii, int*p, int*flag ) ;
void searchinb ( int *sp, int jj, int colofa, int*p, int*flag ) ;
void display_result ( struct sparse ) ;
void delsparse ( struct sparse * ) ;
void main( )
{
struct sparse s[5] ;
int i ;
clrscr( ) ;
for ( i = 0 ; i<= 3 ; i++ )
initsparse ( &s[i] ) ;
create_array ( &s[0] ) ;
create_tuple ( &s[1], s[0] ) ;
display_tuple ( s[1] ) ;
create_array ( &s[2] ) ;
create_tuple ( &s[3], s[2] ) ;
display_tuple ( s[3] ) ;
prodmat ( &s[4], s[1], s[3] ) ;
printf ( "\nResult of multiplication of two matrices: " ) ;
display_result ( s[4] ) ;
for ( i = 0 ; i<= 3 ; i++ )
delsparse ( &s[i] ) ;
getch( ) ;
}
/* initialises elements of structure */
void initsparse ( struct sparse *p )
{
p -> sp = NULL ;
p -> result = NULL ;
}
/* dynamically creates the matrix */
void create_array ( struct sparse *p )
{
int n, i ;
/* allocate memory */
p -> sp = ( int * ) malloc ( MAX1 * MAX2 * sizeof ( int ) ) ;
/* add elements to the array */
for ( i = 0 ; i< MAX1 * MAX2 ; i++ )
{
printf ( "Enter element no. %d: ", i ) ;
scanf ( "%d", &n ) ;
* ( p -> sp + i ) = n ;
}
}
/* displays the contents of the matrix */
```

```
{
int i ;
/* traverses the entire matrix */
for ( i = 0 ; i< MAX1 * MAX2 ; i++ )
{
/* positions the cursor to the new line for every new row */
if ( i % 3 == 0 )
printf ( "\n" ) ;
printf ( "%d\t", * ( s.sp + i ) ) ;
}
}
/* counts the number of non-zero elements */
int count ( struct sparse s )
{
int cnt = 0, i ;
for ( i = 0 ; i< MAX1 * MAX2 ; i++ )
{
if ( * ( s.sp + i ) != 0 )
cnt++ ;
}
return cnt ;
}
/* creates an array that stores information about non-zero elements */
void create_tuple ( struct sparse *p, struct sparse s )
{
int r = 0 , c = -1, l = -1, i ;
/* get the total number of non-zero elements */
p -> row = count ( s ) + 1 ;
/* allocate memory */
p -> sp = ( int * ) malloc ( p -> row * 3 * sizeof ( int ) ) ;
/* store information about total no. of rows, cols, and non-zero values */
* ( p -> sp + 0 ) = MAX1 ;
* ( p -> sp + 1 ) = MAX2 ;
* ( p -> sp + 2 ) = p -> row - 1 ;
l = 2 ;
/* scan the array and store info. about non-zero values in the 3-tuple */
for ( i = 0 ; i< MAX1 * MAX2 ; i++ )
{
c++ ;
/* sets the row and column values */
if ( ( ( i % 3 ) == 0 ) && ( i != 0 ) )
{
r++ ;
c = 0 ;
}
/* checks for non-zero element, row, column and non-zero value is assigned to the matrix */
if ( * ( s.sp + i ) != 0 )
{
l++ ;
* ( p -> sp + l ) = r ;
l++ ;
* ( p -> sp + l ) = c ;
l++ ;
```

```
}
}
}
/* displays the contents of the matrix */
void display_tuple ( struct sparse s )
{
int i, j ;
/* traverses the entire matrix */
printf ( "\nElements in a 3-tuple: " ) ;
j = ( * ( s.sp + 2 ) * 3 ) + 3 ;
for ( i = 0 ; i< j ; i++ )
{
/* positions the cursor to the new line for every new row */
if ( i % 3 == 0 )
printf ( "\n" ) ;
printf ( "%d\t", * ( s.sp + i ) ) ;
}
printf ( "\n" ) ;
}
/* performs multiplication of sparse matrices */
void prodmat ( struct sparse *p, struct sparse a, struct sparse b )
{
int sum, k, position, posi, flaga, flagb, i , j ;
k = 1 ;
p -> result = ( int * ) malloc ( MAXSIZE * 3 * sizeof ( int ) ) ;
for ( i = 0 ; i< * ( a.sp + 0 * 3 + 0 ) ; i++ )
{
for ( j = 0 ; j< * ( b.sp + 0 * 3 + 1 ) ; j++ )
{
/* search if an element present at ith row */
searchina ( a.sp, i, &position, &flaga ) ;
if ( flaga == TRUE )
{
sum = 0 ;
/* run loop till there are element at ith row in first 3-tuple */
while ( * ( a.sp + position * 3 + 0 ) == i )
{
/* search if an element present at ith col. in second 3-tuple */
searchinb ( b.sp, j, * ( a.sp + position * 3 + 1 ), &posi, &flagb ) ;
/* if found then multiply */
if ( flagb == TRUE )
sum = sum + * ( a.sp + position * 3 + 2 ) * * ( b.sp + posi * 3 + 2 ) ;
position = position + 1 ;
}
/* add result */
if ( sum != 0 )
{
* ( p -> result + k * 3 + 0 ) = i ;
* ( p -> result + k * 3 + 1 ) = j ;
* ( p -> result + k * 3 + 2 ) = sum ;
k = k + 1 ;
}
}
```

```
}
/* add total no. of rows, cols and non-zero values */
* ( p -> result + 0 * 3 + 0 ) = * ( a.sp + 0 * 3 + 0 ) ;
* ( p -> result + 0 * 3 + 1 ) = * ( b.sp + 0 * 3 + 1 ) ;
* ( p -> result + 0 * 3 + 2 ) = k - 1 ;
}
/* searches if an element present at iith row */
void searchina ( int *sp, int ii, int *p, int *flag )
{
int j ;
*flag = FALSE ;
for ( j = 1 ; j<= * ( sp + 0 * 3 + 2 ) ; j++ )
{
if ( * ( sp + j * 3 + 0 ) == ii )
{
*p = j ;
*flag = TRUE ;
return ;
}
}
}
/* searches if an element where col. of first 3-tuple is equal to row of second 3-tuple */
void searchinb ( int *sp, int jj, int colofa, int *p, int *flag )
{
int j ;
*flag = FALSE ;
for ( j = 1 ; j<= * ( sp + 0 * 3 + 2 ) ; j++ )
{
if ( * ( sp + j * 3 + 1 ) == jj && * ( sp + j * 3 + 0 ) == colofa )
{
*p = j ;
*flag = TRUE ;
return ;
}
}
}
/* displays the contents of the matrix */
void display_result ( struct sparse s )
{
int i ;
/* traverses the entire matrix */
for ( i = 0 ; i< ( * ( s.result + 0 + 2 ) + 1 ) * 3 ; i++ )
{
/* positions the cursor to the new line for every new row */
if ( i % 3 == 0 )
printf ( "\n" ) ;
printf ( "%d\t", * ( s.result + i ) ) ;
}
}
/* deallocates memory */
void delsparse ( struct sparse *s )
{
if ( s -> sp != NULL )
```

```
if ( s -> result != NULL )
free ( s -> result ) ;
}
```

Input:

First matrices

```
[0 2 3]
[4 0 0]
[0 0 5]
```

Second matrices

```
[0 0 7]
[0 8 0]
[0 9 6]
```

Output:

```
[0 43 18]
[0  0 28]
[0 45 30]
```

Lab assignment:

1) Write a program to add two sparse matrices.

Exercise No. 8: Recursive and Non-recursive traversal of Trees

Description:

A binary tree is a data structure that is defined as a collection of elements called nodes. In a binary tree, the topmost element is called the root node, and each node has 0, 1, or at the most 2 children. A node that has zero children is called a leaf node or a terminal node. Every node contains a data element, a left pointer which points to the left child, and a right pointer which points to the right child. The root element is pointed by a `'root'` pointer. If `root = NULL`, then it means the tree is empty.

Aim: Write a program to implement a binary tree using recursion.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
struct node
{
int data;
struct node *left,*right;
};
struct node *root;
void insert(int x)
{
  struct node *p,*previous,*current;
  p=(struct node *)malloc(sizeof(struct node));
  if(p==NULL)
  {
   printf("\n Out of memory");
  }
  p->data=x;
  p->left=NULL;
  p->right=NULL;
  if(root=NULL)
  {
   root=p;
   return;
  }
  previous=NULL;
  current=root;
  while(current!=NULL)
  {
     previous=current;
     if(p->data<current->data)
```

```
            current=current->right;
            }
     if(p->data<previous->data)
         previous->left=p;
     else
 previous->right=p;
}
void inorder(struct node *t)
{
  if (t!=NULL)
  {
  inorder(t->left);
  printf("\n %5d",t->data);
  inorder (t->right);
  }
}
void del(int x)
{
   int tright=0,tleft=0;
   struct node *ptr=root;
   struct node *parent=root;
   struct node *t1=root;
   struct node *temp=root;
   while(ptr!=NULL&& ptr->data!=x)
   {
      parent=ptr;
      if (x<ptr->data)
           ptr=ptr->left;
      else
          ptr=ptr->right;
   }
   if (ptr==NULL)
   {
     printf("\n Delete element not found");
     return ;
   }
   else if(t1->data==x && (t1->left ==NULL || t1->right==NULL))
         if(t1->left==NULL)
           t1=t1->right;
         else
           t1=t1->left;
   else if (ptr->left==NULL)
     if (x<parent->data)
       parent->left=ptr->right;
     else
       parent->right=ptr->right;
   else if (ptr->right==NULL)
     if (x<parent->data)
       parent->left=ptr->left;
     else
       parent->right=ptr->left;
   else
   {
   temp=ptr;
   parent=ptr;
   if((ptr->left)>=(ptr->right))
   {
      ptr=ptr->left;
      while(ptr->right!=NULL)
      {
          tright=1;
```

```
    }
    temp->data=ptr->data;
    if(tright)
        parent->right=ptr->left;
    else
        parent->left=ptr->left;
  }
 else
  {
   ptr=ptr->right;
   while (ptr->left!=NULL)
   {
        tleft=1;
        parent=ptr;
        ptr=ptr->left;
    }
    temp->data=ptr->data;
    if(tleft)
        parent->left=ptr->right;
    else
        parent->right=ptr->right;
  }
  free(ptr);
 }
}

void main()
{
int op,n,srchno;
root=(struct node *)malloc(sizeof(struct node));
root->data=30;
root->right=root->left=NULL;
clrscr();
do
{
   printf("\n 1.Insertion");
   printf("\n 2.Deletion");
   printf("\n 3.Inorder");
   printf("\n 4.Quit");
   printf("\n Enter your choice\n");
   scanf("%d",&op);

 switch (op)
  {
   case 1: printf("\n Enter the element to insert\n");
           scanf("%d",&n);
           insert(n);
           break;
   case 2: printf("\n Enter the element to be deleted\n");
           scanf("%d",&srchno);
           del(srchno);
           break;
   case 3: printf("\n The inorder elements are\n");
           inorder(root);
           getch();
           break;
   default: exit(0);
  }
 }while(op<4);
 getch();

}
```

1 2 3
Output:
Enter the element to insert1
Enter the element to insert2
Enter the element to insert3
The inorder elements are
2 1 3
Lab assignment:

1) Write a program to implement a binary tree without using recursion

Exercise No. 9: AVL tree implementation

Description:

An AVL tree is the same as that of a binary search tree but with a little difference.
In its structure, it stores an additional variable called theBalance Factor. Thus, every node has a balance factor associated with it. The balance factor of a node is calculated by subtracting the height of its right sub-tree from the height of its left sub-tree. A binary search tree in which every node has a balance factor of –1, 0, or 1 is said to be height balanced. A node with any other balance factor is considered to be unbalanced and requires rebalancing of the tree.
Balance factor = Height (left sub-tree) – Height (right sub-tree)

Aim: Write a program to implement AVL tree

Program:

```
#include <stdio.h>
typedef enum { FALSE ,TRUE } bool;
struct node
{
 int val;
 int balance;
 struct node *left_child;
 struct node *right_child;
};
struct node* search(struct node *ptr, int data)
{
 if(ptr!=NULL)
   if(data < ptr -> val)
   ptr = search(ptr -> left_child,data);
   else if( data > ptr -> val)
   ptr = search(ptr -> right_child, data);
 return(ptr);
}
struct node *insert (int data, struct node *ptr, int *ht_inc)
{
 struct node *aptr;
struct node *bptr;
if(ptr==NULL)
 {
 ptr = (struct node *) malloc(sizeof(struct node));
 ptr -> val = data;
 ptr -> left_child = NULL;
 ptr -> right_child = NULL;
 ptr -> balance = 0;
 *ht_inc = TRUE;
 return (ptr);
 }
if(data < ptr -> val)
 {
 ptr -> left_child = insert(data, ptr -> left_child, ht_inc);
 if(*ht_inc==TRUE)
  {
   switch(ptr -> balance)
```

```
case -1: /* Right heavy */


ptr -> balance = 0;
   *ht_inc = FALSE;
   break;


case 0: /* Balanced */


ptr -> balance = 1;


break;
   case 1: /* Left heavy */


aptr = ptr -> left_child;
    if(aptr -> balance == 1)
     {


printf("Left to Left Rotation\n");


ptr -> left_child= aptr -> right_child;


aptr -> right_child = ptr;


ptr -> balance = 0;


aptr -> balance=0;
     ptr = aptr;
     }
    else
     {
     printf("Left to right rotation\n");


bptr = aptr -> right_child;
     aptr -> right_child = bptr -> left_child;


bptr -> left_child = aptr;


ptr -> left_child = bptr -> right_child;


bptr -> right_child = ptr;


if(bptr -> balance == 1 )


pt
```

```
else


pt
r -> balance = 0;


if(bptr -> balance == -1)


aptr -> balance = 1;


else


aptr -> balance = 0;


bptr -> balance=0;


ptr = bptr;
     }
    *ht_inc = FALSE;
    }
  }
 }
if(data > ptr -> val)
{
  ptr -> right_child = insert(info, ptr -> right_child, ht_inc);
  if(*ht_inc==TRUE)
  {
   switch(ptr -> balance)
   {


case 1: /* Left heavy */
     ptr -> balance = 0;
     *ht_inc = FALSE;
     break;


case 0: /* Balanced */
     ptr -> balance = -1;
     break;
    case -1: /* Right heavy */

 aptr = ptr -> right_child;
  if(aptr -> balance == -1)
     {
    printf("Right to Right Rotation\n");
  ptr -> right_child= aptr -> left_child;
    aptr -> left_child = ptr;
      ptr -> balance = 0;
     aptr -> balance=0;
  ptr = aptr;
     }
  else
     {
```

```
  aptr -> left_child = bptr -> right_child;
bptr -> right_child = aptr;
ptr -> right_child = bptr -> left_child;
  bptr -> left_child = pptr;
if(bptr -> balance == -1)
ptr -> balance = 1;
      else
ptr -> balance = 0;
      if(bptr -> balance == 1)
  aptr -> balance = -1;
      else
  aptr -> balance = 0;
      bptr -> balance=0;
  ptr = bptr;
}/*End of else*/
*ht_inc = FALSE;
   }
  }
 }
 return(ptr);
}
void display(struct node *ptr, int level)
{
 int i;
 if ( ptr!=NULL )
 {
  display(ptr -> right_child, level+1);
  printf("\n");
  for (i = 0; i < level; i++)
   printf(" ");
  printf("%d", ptr -> val);
  display(ptr -> left_child, level+1);
 }
}
void inorder(struct node *ptr)
{
 if(ptr!=NULL)
 {
  inorder(ptr -> left_child);
  printf("%d ",ptr -> val);
  inorder(ptr -> right_child);
 }
}
main()
{
 bool ht_inc;
 int data ;
 int option;
 struct node *root = (struct node *)malloc(sizeof(struct node));
root = NULL;
while(1)
 {
 printf("1.Insert\n");
 printf("2.Display\n");
 printf("3.Quit\n");
 printf("Enter your option : ");
 scanf("%d",&option);
 switch(choice)
   {
   case 1:
   printf("Enter the value to be inserted : ");
```

```
    root = insert(data, root, &ht_inc);
 else
 printf("Duplicate value ignored\n");
  break;
 case 2:
   if(root==NULL)
    {
 printf("Tree is empty\n");
    continue;
    }
   printf("Tree is :\n");
   display(root, 1);
 printf("\n\n");
printf("Inorder Traversal is: ");
 inorder(root);
   printf("\n");
   break;
  case 3:
   exit(1);
 default:
 printf("Wrong option\n");
    }
  }
 }
```

Input:

6 11 2 4 3 5

Output:

2 3 5 4 6 11

Lab Assignment:

1) Write a program to implement AVL tree

Exercise No. 10: Application of sorting and searching algorithms

Description:

To search an element in an array is known as searching and to sort the element in an ascending and descending order is known as sorting.Two type of searching linear and binary. Mainly five type of sorting like bubble ,insertion ,selection, merge and quick sort.here we mainly focus on binary search and merge and quick sort.

Aim:Implement Binary search without using recursion

Program:

```
#include<stdio.h>

int main(){

   int a[10],i,n,m,c=0,l,u,mid;

   printf("Enter the size of an array: ");
   scanf("%d",&n);

   printf("Enter the elements in ascending order: ");
   for(i=0;i<n;i++){
       scanf("%d",&a[i]);
   }

   printf("Enter the number to be search: ");
   scanf("%d",&m);

   l=0,u=n-1;
   while(l<=u){
       mid=(l+u)/2;
```

```
            c=1;
            break;
        }
        else if(m<a[mid]){
            u=mid-1;
        }
        else
            l=mid+1;
    }
    if(c==0)
        printf("The number is not found.");
    else
        printf("The number is found.");

    return 0;
}
```

OUTPUT:

```
Enter the size of an array: 5
        Enter the element in ascending order: 2  4  8  9  12
        Enter the number to be  search: 3
        The number is not found.
```

Aim: Implement Merge Sort using Divide and Conquer approach

Program:

```
#include<stdio.h>
#include<conio.h>
void merge(int [],int ,int ,int );
void part(int [],int ,int );
int main()
{
 int arr[30];
 int i,size;
 printf("\n\t------- Merge sorting method -------\n\n");
 printf("Enter total no. of elements : ");
 scanf("%d",&size);
 for(i=0; i<size; i++)
 {
   printf("Enter %d element : ",i+1);
   scanf("%d",&arr[i]);
 }
 part(arr,0,size-1);
 printf("\n\t------- Merge sorted elements -------\n\n");
 for(i=0; i<size; i++)
 printf("%d ",arr[i]);
 getch();
 return 0;
}

void part(int arr[],int min,int max)
{
 int mid;
 if(min<max)
```

```
  mid=(min+max)/2;
  part(arr,min,mid);
  part(arr,mid+1,max);
  merge(arr,min,mid,max);
 }
}

void merge(int arr[],int min,int mid,int max)
{
  int tmp[30];
  int i,j,k,m;
  j=min;
  m=mid+1;
  for(i=min; j<=mid && m<=max ; i++)
  {
    if(arr[j]<=arr[m])
    {
       tmp[i]=arr[j];
       j++;
    }
    else
    {
       tmp[i]=arr[m];
       m++;
    }
  }
  if(j>mid)
  {
    for(k=m; k<=max; k++)
    {
       tmp[i]=arr[k];
       i++;
    }
  }
  else
  {
    for(k=j; k<=mid; k++)
    {
      tmp[i]=arr[k];
      i++;
    }
  }
  for(k=min; k<=max; k++)
    arr[k]=tmp[k];
}
```

Output:

```
Enter the no of elements:7
7 8 9 4 5 3 1
The unsorted list is: 7 8 9 4 5 3 1
```

1 3 4 5 7 8 9

Aim:Implement Quick Sort using Divide and Conquer approach
Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define MAX 6000

void quick(int x[],int lb,int ub);
int partition(int x[],int lb,int ub);

void main()
{
        int i,n,x[MAX];
        time_t start,end;
        clrscr();
        printf("Enter the number of elements: ");
        scanf("%d",&n);

        for(i=0;i<n;i++)
                x[i]=rand();

        printf("\nEntered array is \n");
        for(i=0;i<n;i++)
                printf("%d ",x[i]);

        start=time(NULL);
        quick(x,0,n-1);
        end=time(NULL);
        printf("Sorted array is as shown:\n");
        for(i=0;i<n;i++)
                printf("%d ",x[i]);
        printf("\nTIME for %d elements : %f", n, difftime(end,start));
        getch();

}

void quick(int x[],int lb,int ub)
{
        int j;
        if(lb<ub)
        {
                printf("\n");
                j=partition(x,lb,ub);
                quick(x,lb,j-1);
                quick(x,j+1,ub);
        }
}
```

```
{
        int a,down,up,temp;
        a=x[lb];
        up=ub;
        down=lb;
        while(down<up)
        {
                while(x[down]<=a&&down<ub)
                        down++;
                while(x[up]>a)
                        up--;
                if(down<up)
                {
                        temp=x[down];
                        x[down]=x[up];
                        x[up]=temp;
                }
        }
        x[lb]=x[up];
        x[up]=a;
        return up;
}
```

Output:

```
Enter the number of elements:5
    Entered array is
      41   18467    6334    26500    19169
    Sorted array is as shown
      41   6334     18467    19169    26500
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Computer Organization Lab
Course Code: CS394
L-T-P Scheme: 0-0-3 **Course Credits: 2**

Objective:
1. Understand the architecture of a modern computer with its various processing units.
2. To learn and understand IC of basic gates.
3. To provide an efficient understanding of the Hardware, design complete circuit.

Learning Outcomes: The students will have a detailed knowledge of the concept of IC
1. Students can understand the architecture of modern computer.
2. They can analyze the Performance of a computer using performance equation
3. Students can calculate the effective address of an operand by addressing modes
4. They can understand how computer stores positive and negative numbers.
5. Understanding of how a computer performs arithmetic operation of positive and negative numbers.
6. Understanding of how computer stores floating point numbers in IEEE 754 standard.
7. Students can understand how cache mapping occurs in computer and can solve various problems related to this.
8. Secondary storage organization and problem solving

Course Contents:

Unit –I: Basic gates

Study about logic gates and verify their truth tables. XOR (IC 7486), OR (IC 7432),NOT (IC 7404), AND (IC 7408),NAND (IC 7400), etc. Also implementation basic gates using universal gate (NAND).

Unit –II: Half adder, Full Adder

Implement Half and Full Adder using basic gates and check with the following truth table. Half Adder and Full Adder circuits is explained with their truth tables in this article. Design of Full Adder using Half Adder circuit is also shown. Single-bit Full Adder circuit and Multi-bit addition using Full Adder

Unit –III: Half Substractor, Full Substractor.

Implement Half and Full Adder using basic gates and check with the following truth table. Half Subtractor is used for subtracting one single bit binary digit from another single bit binary digit. Full Subtractor, A logic Circuit Which is used for Subtracting Three Single bit Binary digit is known as Full Subtractor

Unit –IV: 4-bit parallel Binary adder and substractor.

The arithmetic addition of two binary digits, together with an input carry from a previous stage. The serial addition method uses only one full-adder circuit and a storage device to hold the generated output carry and sum.

Unit –V: BCD adder

The arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot be greater than 19, the 1 in the sum being an input carry.

Unit –VI: 8 to 1 Multiplexer unit (MUX)
It transfer a large number of information units over a smaller number of channels, (usually one channel) under the control of selection signals. Multiplexer means many to one. A multiplexer is a circuit with many inputs but only one output.

Unit –VII: DEMULTIPLEXER
It perform the opposite function of multiplexers.

Unit –VIII: BCD to 7 segment decoder
Using digital kit implement Digital number (0,1,2,3,4,5,6,7,8,9)

Unit –IX: BCD TO EXCESS 3CODE CONVERTOR

The excess-3 code digit is obtained by adding three to the corresponding BCD digit.

Unit –X: FLIP FLOP
S-R Flip Flop, J-K Flip Flop, T Flip Flop, T Flip Flop

Unit –X: Design a composite ALU.

Implement Airthmatic Logic Unit Arithmetic operations are like addition ,substraction, multiplication, and division. Logical operations are like and, or nand, nor ,not operations on bits

Text Book:
1. David A. Patterson, John L. Hennessy, “Computer Organization and Design”, Elsevier.

References:
1. S.Salivahanan & S.Arivazhagan, **“**Digital Circuits and Design”, VIKAS publishing house PVT LTD

Recommended Systems/ Software Requrements:
1. Trainer kit
2. IC (Integrated Circuit)
3. Wire/ Probes

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

LIST OF EXPERIMENTS

1. Realization of the basic gates (AND, OR, NOT) and universal gates (NAND, NOR).
2. Design and implementation of basic gates using universal gate (NAND).
3. Design and implementation of half adder.
4. Design and implementation of full adder.
5. Design and implementation of half substractor.
6. Design and implementation of full substractor.
7. Design of a 4-bit parallel Binary adder circuit using the IC-Chip 7483.
8. Design of an Adder/Subtractor composite unit circuit using the IC-Chip 7483.
9. Design a BCD adder using two 7483 IC chip.
10. Design an 8 to 1 Multiplexer unit (MUX) using basic gates and using IC 74151
11. Design an 8 to 1 Multiplexer unit (MUX) using basic gates and using IC 74153.
12. Design and implementation of DEMULTIPLEXER .
13. Design of a BCD to 7 segment decoder.
14. Design and implementation of BCD TO EXCESS 3CODE CONVERTOR
15. Design and implementation of SR LATCH ,SR FLIP FLOP AND JK FLIP FLOP.
16. Use a multiplexer unit to design a composite ALU.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT NO.-1

AIM:- To study about logic gates and verify their truth tables.

APPARATUS REQUIRED: IC 7486, IC 7432, IC 7408, IC 7400, etc.

THEORY:

Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output. OR, AND and NOT are basic gates. NAND, NOR and X-OR are known as universal gates. Basic gates form these gates.

AND GATE:

The AND gate performs a logical multiplication commonly known as AND function. IC 7408 The output is high when both the inputs are high. The output is low level when any one of the inputs is low.

OR GATE:

The OR gate performs a logical addition commonly known as OR function. IC 7432. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.

NOT GATE:

The NOT gate is called an inverter. IC 7404 The output is high when the input is low. The output is low when the input is high.

NAND GATE:

The NAND gate is a contraction of AND-NOT. IC 7400. The output is high when both inputs are low and any one of the input is low .The output is low level when both inputs are high.

NOR GATE:

The NOR gate is a contraction of OR-NOT. IC7402. The output is high when both inputs are low. The output is low when one or both inputs are high.

X-OR GATE:

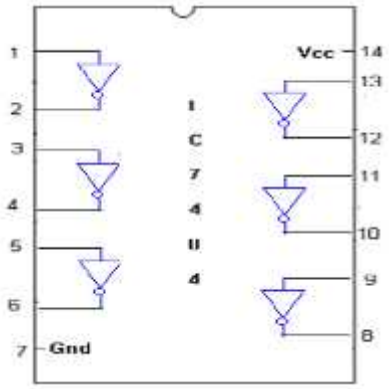
The output is high when any one of the inputs is high. IC 7486 The output is low when both the inputs are low and both the inputs are high.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

LOGIC GATES SYMBOL TRUTH TABLE AND PIN DIAGRAM:

NOT GATE

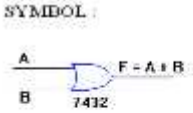


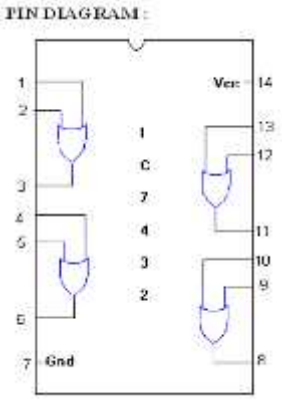
TRUTH TABLE :

A	$\overline{A}$
0	1
1	0

OR GATE:

SYMBOL: **PIN DIAGRAM:**

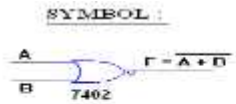


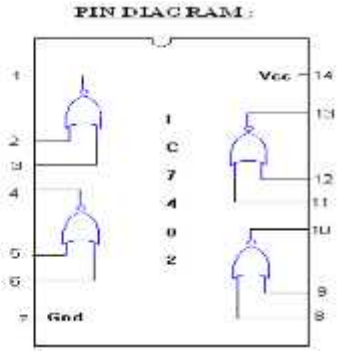


TRUTH TABLE

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOR GATE:





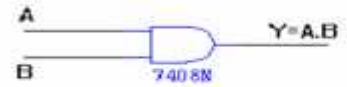
TRUTH TABLE

A	B	$\overline{A+B}$
0	0	1
0	1	1
1	0	1
1	1	0

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

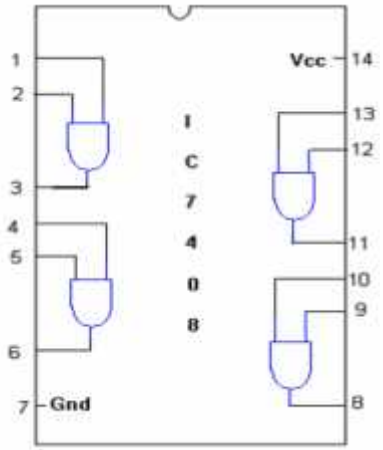
Lab Manual

AND GATE:



TRUTH TABLE

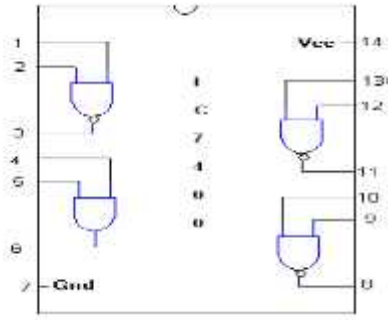
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1



NAND GATE:

TRUTH TABLE

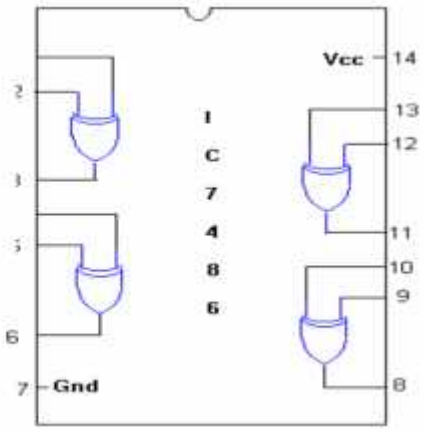
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0



EX-OR GATE:

TRUTH TABLE :

A	B	$\overline{A}B + A\overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0



PROCEDURE:

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

CONCLUSION:

A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels. The logic state of a terminal can, and generally does, change often, as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V).

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

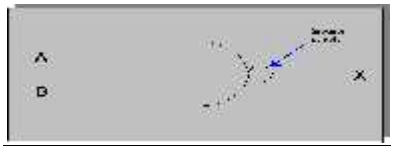
EXPERIMENT NO:2

AIM: Design and implementation of basic gates using universal gate (NAND).

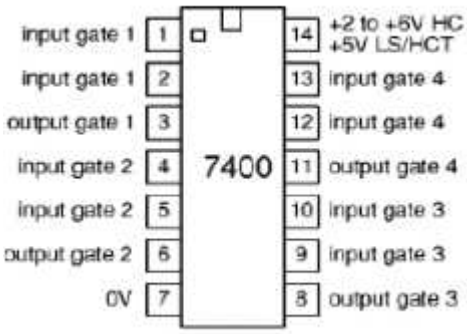
APPRATUS REQUIRED: IC 7400

THEORY: The NAND Gate:

The NAND, which is composed of two or more inputs and a single output, is a very popular logic element because it may be used as a universal function. That is, it may be employed to construct an inverter, an AND gate, an OR gate, or any combination of theses functions. The term NAND is formed by the concatenation NOT-AND and implies an AND function with an inverted output. The standard symbol for the NAND gate is shown in Figure 1-7 and its truth table listed in Table 1-4. The logical operation of the NAND gate is such that the output is LOW (0) only when all the inputs are HIGH (1).



PIN DIAGRAM:



Circuit Diagrams

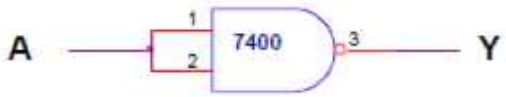
INPUT		OUTPUT
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

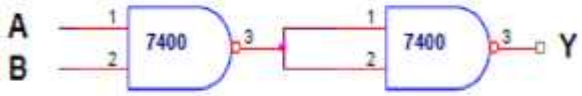
Not gate



INPUT A	OUTPUT Y
0	1
1	0

Truth Table

AND gate



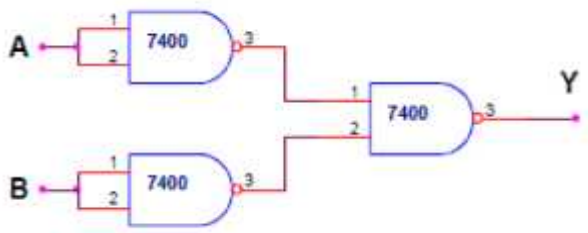
INPUT		OUTPUT Y
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

OR gate

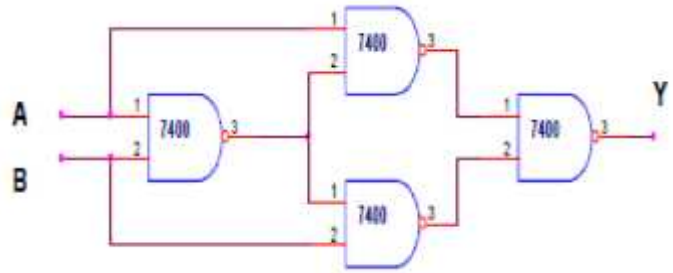
INPUT		OUTPUT Y
A	B	
0	0	0
0	1	1
1	0	1

1	**1**	**1**

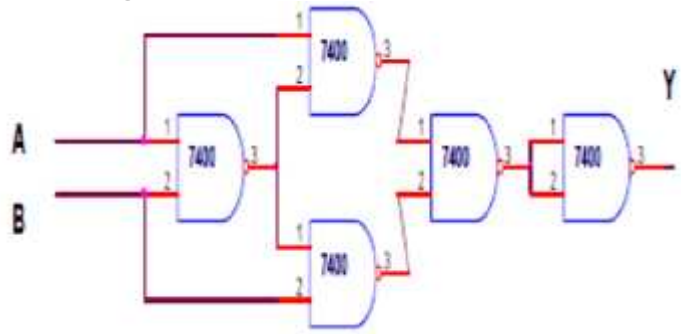


INPUT		OUTPUT
A	**B**	**Y**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Ex-OR gate



Ex-NOR gate



INPUT		OUTPUT
A	**B**	**Y**
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

PROCEDURE:

1. Connect the logic gates as shown in the diagrams using IC 7400 NAND gate.

2. Feed the logic signals 0 or 1 from the logic input switches in different combinations at the inputs A & B.
3. Monitor the output using logic output LED indicators.
4. Repeat steps 1 to 3 for NOT, AND, OR, EX – OR & EX-NOR operations and compare the outputs with the truth tables.

CONCLUSION:

A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels. The logic state of a terminal can, and generally does, change often, as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V).

PRECAUTIONS:

1. All the connections should be made properly.
2. IC should not be reversed.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT NO:3

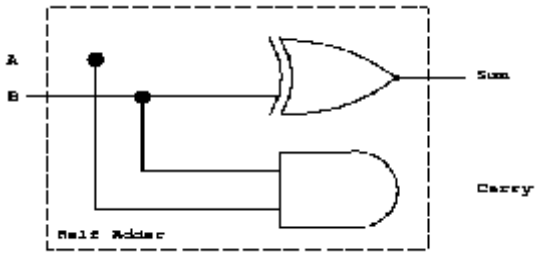
AIM: Design and implementation of HALF ADDER.

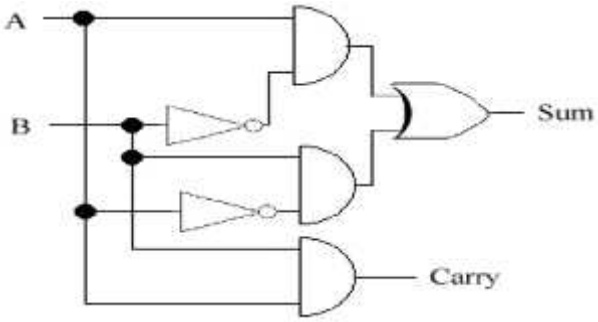
APPRATUS REQUIRED: IC 7486, IC 7432, IC 7408, IC 7404, IC 7400, etc.

THEORY:

Half-Adder: A combinational logic circuit that performs the addition of two data bits, A and B, is called a half-adder. Addition will result in two output bits; one of which is the sum bits, and the other is the carry bit, C.

CIRCUIT DIAGRAM:





TRUTH TABLE:

Truth Table of a Half-Adder			
Inputs		Outputs	
A	B	Sum S	Carry C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logical Expression:-

$S_1 = \bar{A}B + A\bar{B} = A \oplus B$

carry $C_o = A$

PROCEDURE:

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.

3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

CONCLUSION:

Thus, for example, a binary input of 101 results in an output of 1 + 0 + 1 = 10 (decimal number 2). The carry-out represents bit one of the result, while the sum represents bit zero. Likewise, a half adder can be used as a 2:2 lossy compressor, compressing four possible inputs into three possible outputs.

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT NO:4

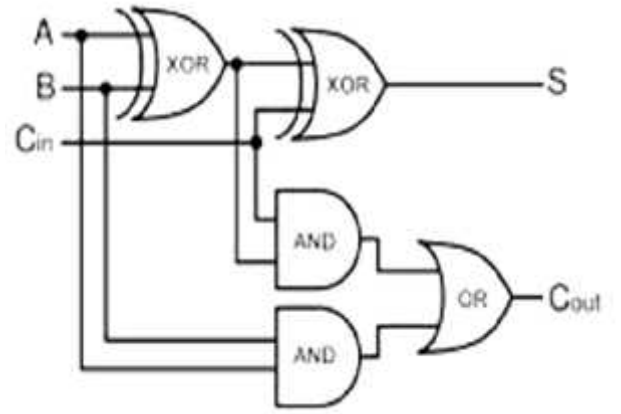
AIM: Design and implementation of FULL ADDER.

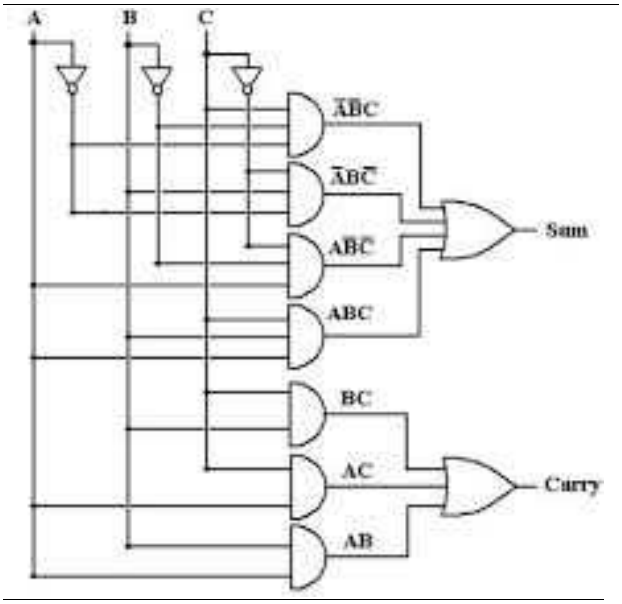
APPRATUS REQUIRED: IC 7486, IC 7432, IC 7408, IC 7404, IC 7400, etc.

THEORY:

Full-Adder*:* The half-adder does not take the carry bit from its previous stage into account. This carry bit from its previous stage is called carry-in bit. A combinational logic circuit that adds two data bits, A and B, and a carry-in bit, Cin , is called a full-adder.

CIRCUIT DIAGRAM:





TRUTH TABLE:

Truth Table of a Full-Adder				
Inputs			Outputs	
Augend Bit A	Addend Bit B	Carry input C_{in}	Sum S	Carry C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

BOOLEAN EXPRESSIONS:
$S = A \oplus B \oplus C$
C=A B + B Cin + A Cin

PROCEDURE:
1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

CONCLUSION:
Thus, for example, a binary input of 101 results in an output of 1 + 0 + 1 = 10 (decimal number 2). The carry-out represents bit one of the result, while the sum represents bit zero.

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT NO:5

AIM: Design and implementation of HALF SUBSTRACTOR.

APPRATUS REQUIRED: IC 7486, IC 7432, IC 7408, IC 7404, IC 7400, etc.

THEORY:

Half Subtractor*:* Subtracting a single-bit binary value B from another A (i.e. A -B) produces a difference bit D and a borrow out bit B-out. This operation is called half subtraction and the circuit to realize it is called a half subtractor.

CIRCUIT DIAGRAM:

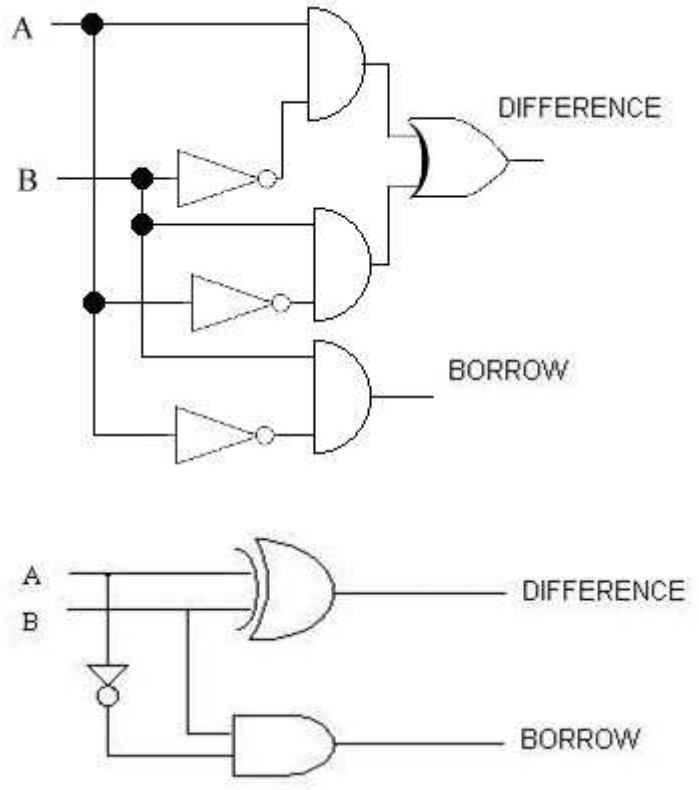


Fig: half substactor.

TRUTH TABLE:

Truth Table of a Half-Subtractor			
Inputs		**Outputs**	
Minuend A	**Subtrahend B**	**Difference D**	**Borrow Bout**
0	**0**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**1**	**0**
1	**1**	**0**	**0**

BOOLEAN EXPRESSION:

$D \quad ., D = \bar{A}B + A\bar{B} = A \oplus B$

$B_o \quad = \bar{A}B$

PROCEDURE:

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

CONCLUSION:

The Binary Subtractor is another type of combinational arithmetic circuit that is the opposite of the Binary Adder we looked at in a previous tutorial. As their name implies, a Binary Subtractor is a decision making circuit that subtracts two binary numbers from each other, for example, X – Y to find the resulting difference between the two numbers.

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

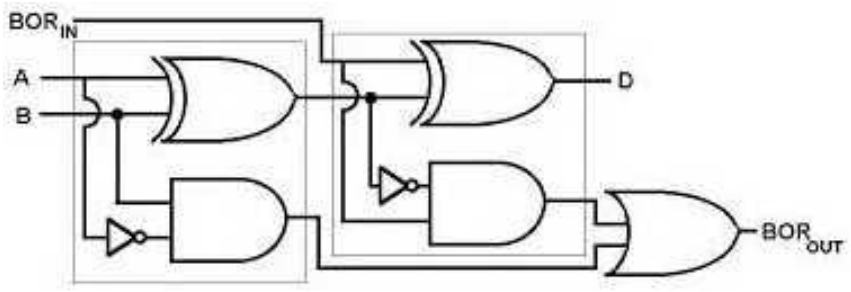
EXPERIMENT NO:6

AIM: Design and implementation of FULL SUBSTRACTOR.

APPRATUS REQUIRED: IC 7486, IC 7432, IC 7408, IC 7404, IC 7400, etc.

THEORY: Full Subtractor: Subtracting two single-bit binary values, B, Cin from a single-bit value A produces a difference bit D and a borrow out Br bit. This is called full subtraction.

CIRCUIT DIAGRAM:



TRUTH TABLE:

Truth Table of a Full-Subtractor				
Inputs			Outputs	
Minuend Bit A	Subtrahend Bit B	Borrow input B_{in}	Difference D	Borrow Out B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

BOOLEAN EXPRESSION: $D = \overline{A}\overline{B}B_{in} + \overline{A}B\overline{B_{in}} + A\overline{B}\overline{B_{in}} + ABB_{in} = A \oplus B \oplus B_{in}$

$B_o = \overline{A}BB_{in} + \overline{A}B\overline{B_{in}} + \overline{A}\overline{B}B_{in} + ABB_{in} = \overline{A}B + \overline{A}B_{in} + BB_{in}$

$= \overline{A}B + \overline{(A \oplus B)}B_{in}$

PROCEDURE:

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

CONCLUSION:

The Binary Subtractor is another type of combinational arithmetic circuit that is the opposite of the Binary Adder we looked at in a previous tutorial. As their name implies, a Binary Subtractor is a decision making circuit that subtracts two binary numbers from each other, for example, X – Y to find the resulting difference between the two numbers.

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

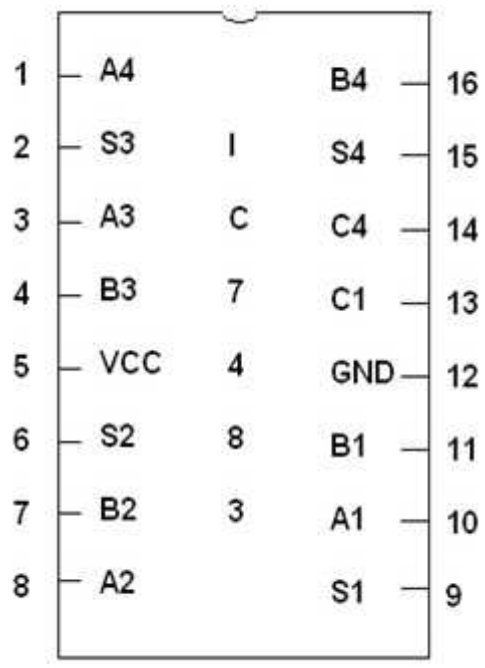
EXPERIMENT NO-07

AIM: Design of a 4-bit parallel Binary adder circuit using the IC-Chip 7483.

APPRATUS REQUIRED: IC 7483 etc.

THEORY: 4-bit parallel Binary adder circuit using the IC-Chip 7483.
Consider the arithmetic addition of two binary digits, together with an input carry from a previous stage. The serial addition method uses only one full-adder circuit and a storage device to hold the generated output carry and sum. The parallel method uses n full-adder circuit. A binary parallel adder is a digital function that produces the arithmetic sum of two binary numbers in parallel.

PIN DIAGRAM FOR IC 7483:



Here A_1, A_2, A_3, A_4 and B_1, B_2, B_3, B_4 are the 4+4 =8 Input. S_1, S_2, S_3, S_4 are the 4 out put where the sum value is stored. So put any two 4 digit binary number and get the sum of them.

PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

CONCLUSION:

Parallel adder is a combinatorial circuit (not clocked, does not have any memory and feedback) adding every bit position of the operands in the same time.

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

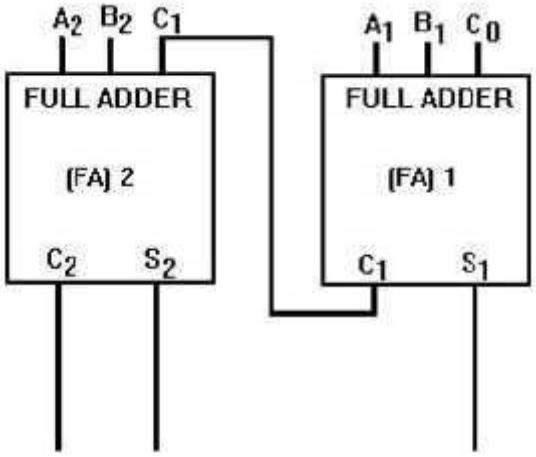
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT NO-08

AIM: 4 Bit binay adder/substractor composite unit.

APPRATUS REQUIRED:

THEORY : The Full adder can add single-digit binary numbers and carries. The largest sum that can be obtained using a full adder is 112. Parallel adders can add multiple-digit numbers. If full adders are placed in parallel, we can add two- or four-digit numbers or any other size desired. Figure below uses STANDARD SYMBOLS to show a parallel adder capable of adding two, two-digit binary numbers The addend would be on A inputs, and the augend on the B inputs. For this explanation we will assume there is no input to C0 (carry from a previous circuit)



To add 102 (addend) and 012 (augend), the addend inputs will be 1 on A2 and 0 on A1. The augend inputs will be 0 on B2 and 1 on B1. Working from right to left, as we do in normal addition, let's calculate the outputs of each full adder. With A1 at 0 and B1 at 1, the output of adder1 will be a sum (S1) of 1 with no carry (C1). Since A2 is 1 and B2 is 0, we have a sum (S2) of 1 with no carry (C2) from adder1. To determine the sum, read the outputs (C2, S2, and S1) from left to right. In this case, C2 = 0, S2 = 1, and S1 = 1. The sum, then, of 102 and 012is 0112. To add four bits we require four full adders arranged in parallel. IC 7483 is a 4- bit parallel adder whose pin diagram is shown.

CIRCUIT DIAGRAM OF ADDER AND SUBSTRACTOR AND 7483 IC PIN DIAGRAM :

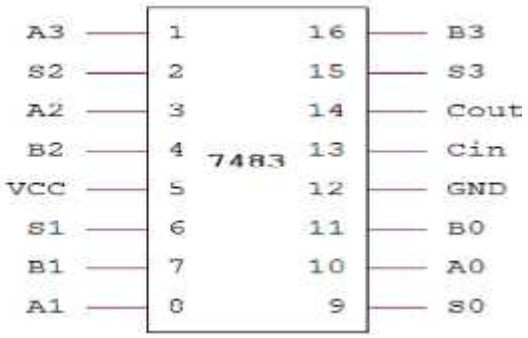


Fig-7483 ic pin diagram.

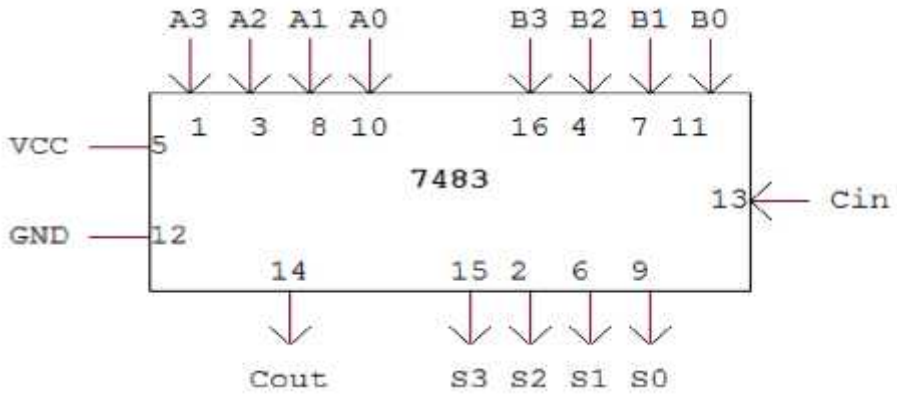


Fig-adder circuit.

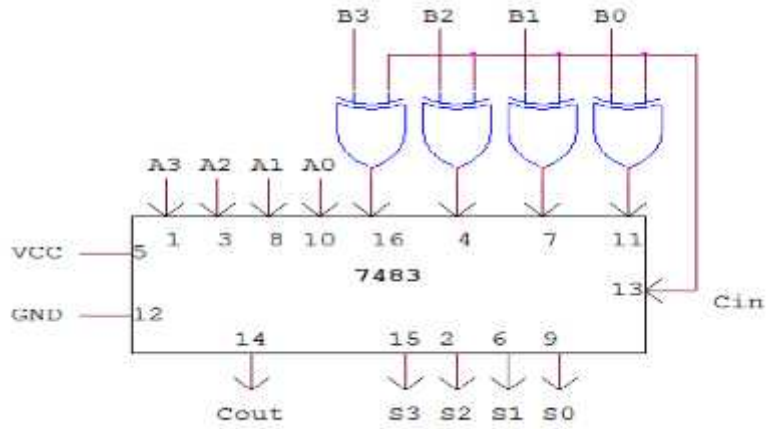
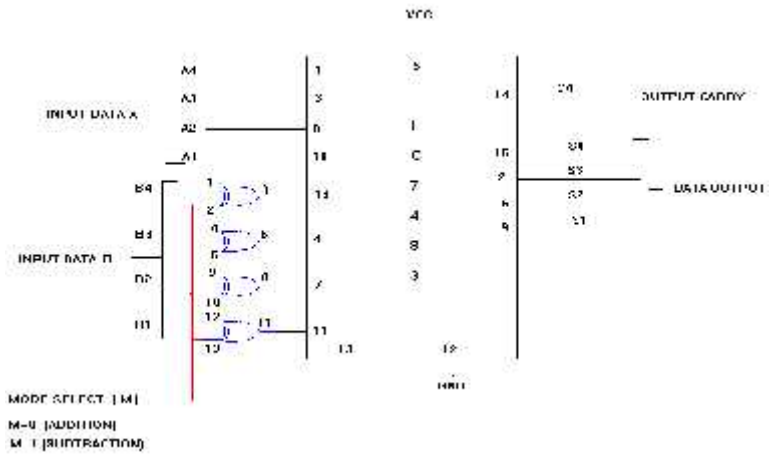


Fig-substractor circuit.

4-BIT BINARY ADDER/SUBTRACTOR

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual



Here A_1, A_2, A_3, A_4 and B_1, B_2, B_3, B_4 are the 8 Input. A_1, A_2, A_3, A_4 data set directly connected to the IC 7483 and B_1, B_2, B_3, B_4 fast connected to the XOR gates then the out put of XOR gates connected to 7483. Now XOR gates other I/P line are connected to the pin no 13 of IC 7483, If the value of pin no 13 is '0' means ADDITION and '1' means SUBTRACTION S_1, S_2, S_3, S_4 are the 4 out put where the sum/ borrow value is stored. So put any two 4 digit binary number and get the sum of them.

PROCEDURE:

1.for adder-

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Apply augend and addend bits on A and B and cin=0.
- Verify the results and observe the outputs.

2.for substractor-

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Apply Minuend and subtrahend bits on A and B and cin=1.
- Verify the results and observe the outputs.

CONCLUSION:

Binary adder is one of the basic combinational logic circuits. The outputs of a combinational logic circuit depend on the present input only. In other words, outputs of combinational logic circuit do not depend upon any previously applied inputs. It does not require any memory like component. Binary adder is one of the basic combinational logic circuits as present state of input variables.

PRECAUTION:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

EXPERIMENT NO-09

AIM: Design and implementation of 4 BCD adder using 7483 ic.

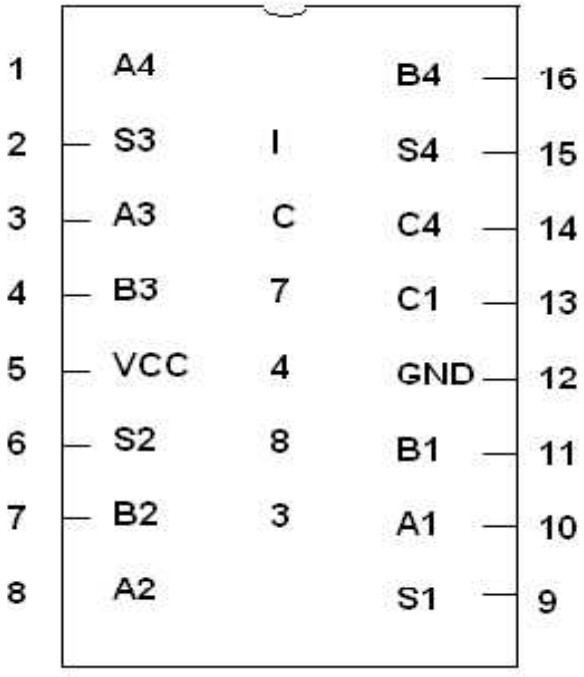
APPRATUS REQUIRED: IC 7483, IC 7432, IC 7408, IC 7400, etc.

THEORY: 4 BIT BCD ADDER:

Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot be greater than 19, the 1 in the sum being an input carry. The output of two decimal digits must be represented in BCD and should appear in the form listed in the columns. A BCD adder that adds 2 BCD digits and produce a sum digit in BCD. The 2 decimal digits, together with the input carry, are first added in the top 4 bit adder to produce the binary sum.

CIRCUIT DIAGRAM:

PIN DIAGRAM FOR IC 7483:

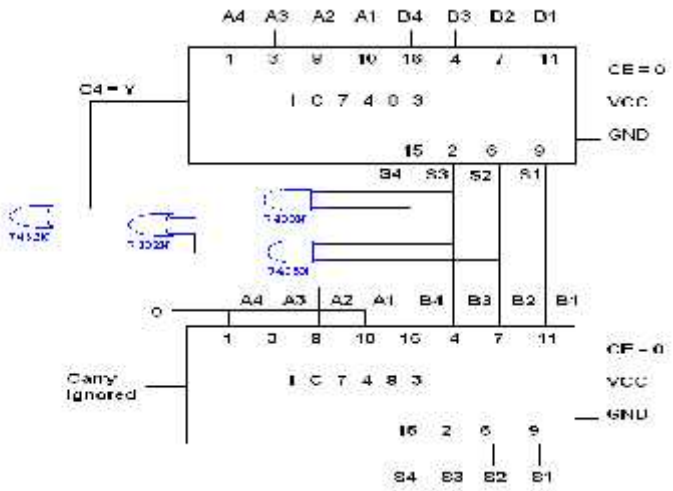


UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

LOGIC DIAGRAM:

BCD ADDER:



TRUTH TABLE:

BCD SUM				CARRY
S4	**S3**	**S2**	**S1**	**C**
0	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**0**
0	**0**	**1**	**0**	**0**
0	**0**	**1**	**1**	**0**
0	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**
0	**1**	**1**	**0**	**0**
0	**1**	**1**	**1**	**0**
1	**0**	**0**	**0**	**0**
1	**0**	**0**	**1**	**0**
1	**0**	**1**	**0**	**1**
1	**0**	**1**	**1**	**1**
1	**1**	**0**	**0**	**1**
1	**1**	**0**	**1**	**1**
1	**1**	**1**	**0**	**1**
1	**1**	**1**	**1**	**1**

PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

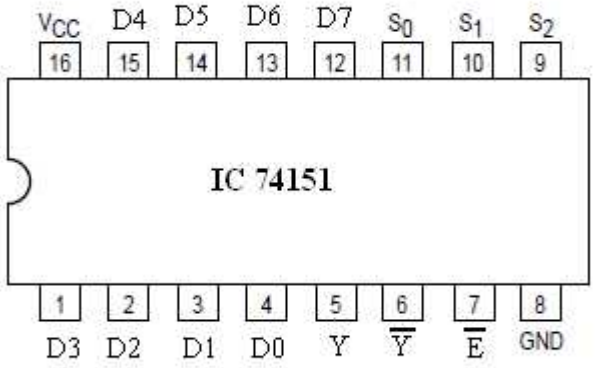
EXPERIMENT NO:10

AIM: Design and implementation of 8:1 MULTIPLEXER .

APPRATUS REQUIRED: IC 74151

THEORY: Multiplexers are very useful components in digital systems. They transfer a large number of information units over a smaller number of channels, (usually one channel) under the control of selection signals. Multiplexer means many to one. A multiplexer is a circuit with many inputs but only one output. By using control signals (select lines) we can select any input to the output. Multiplexer is also called as data selector because the output bit depends on the input data bit that is selected. The general multiplexer circuit has 2n input signals, n control/select signals and 1 output signal.

CIRCUIT DIAGRAM:



TRUTH TABLE:

Select Data Inputs			Output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

CONCLUSION:

The truth table for an 8-to1 multiplexer is given below with eight combinations of inputs so as to generate each output corresponds to input.
For example, if S2= 0, S1=1 and S0=0 then the data output Y is equal to D2. Similarly the data outputs D0 to D7 will be selected through the combinations of S2, S1

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

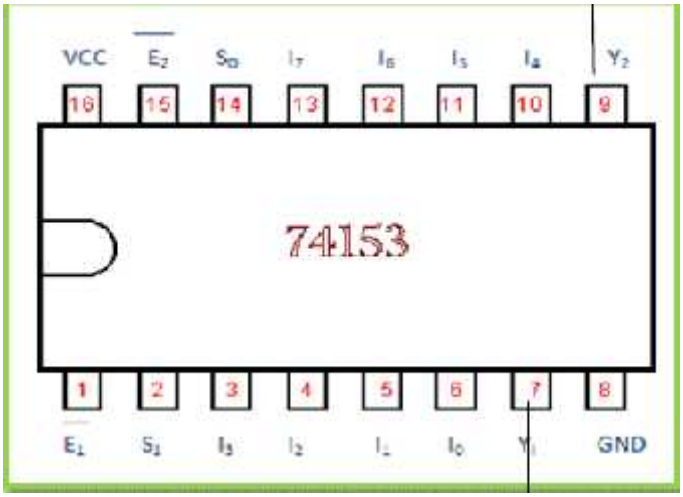
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

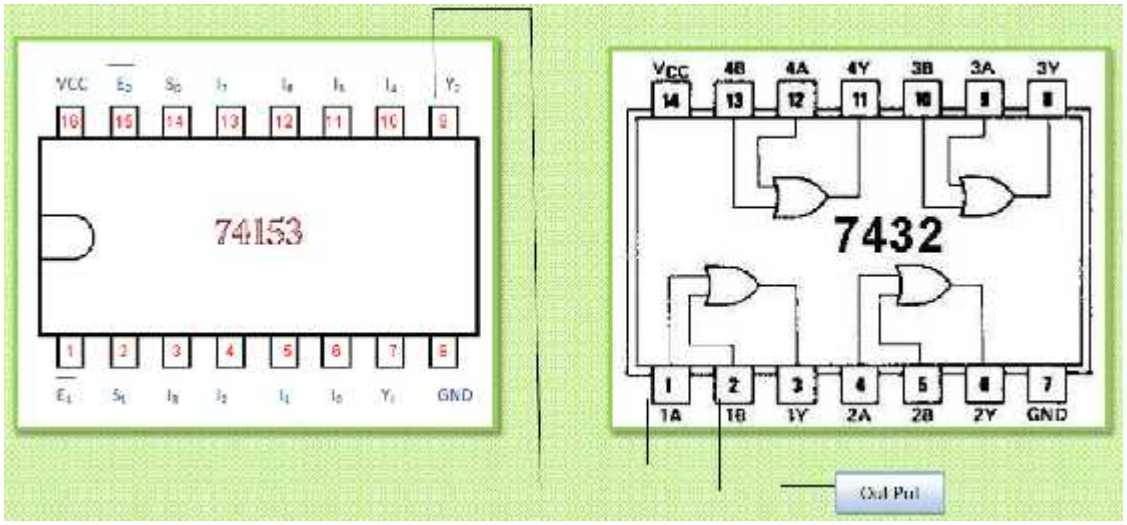
AIM: Design and implementation of 8:1 MULTIPLEXER .

APPRATUS REQUIRED: IC 74153, IC 7432

THEORY: Multiplexers are very useful components in digital systems. They transfer a large number of information units over a smaller number of channels, (usually one channel) under the control of selection signals. Multiplexer means many to one. A multiplexer is a circuit with many inputs but only one output. By using control signals (select lines) we can select any input to the output. Multiplexer is also called as data selector because the output bit depends on the input data bit that is selected. The general multiplexer circuit has 2n input signals, n control/select signals and 1 output signal.

CIRCUIT DIAGRAM:





TRUTH TABLE:

E_1	E_2	S_1	S_0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y (O/P)
1	0	0	0	1	X	X	X	X	X	X	X	ON
1	0	0	0	0	X	X	X	X	X	X	X	OFF
1	0	0	1	X	1	X	X	X	X	X	X	ON
1	0	0	1	X	0	X	X	X	X	X	X	OFF
1	0	1	0	X	X	1	X	X	X	X	X	ON
1	0	1	0	X	X	0	X	X	X	X	X	OFF
1	0	1	1	X	X	X	1	X	X	X	X	ON
1	0	1	1	X	X	X	0	X	X	X	X	OFF
0	1	0	0	X	X	X	X	1	X	X	X	ON
0	1	0	0	X	X	X	X	0	X	X	X	OFF
0	1	0	1	X	X	X	X	X	1	X	X	ON
0	1	0	1	X	X	X	X	X	0	X	X	OFF
0	1	1	0	X	X	X	X	X	X	1	X	ON
0	1	1	0	X	X	X	X	X	X	0	X	ON
0	1	1	1	X	X	X	X	X	X	X	1	ON
0	1	1	1	X	X	X	X	X	X	X	0	OFF

PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

CONCLUSION:

The truth table for an 8-to1 multiplexer is given below with eight combinations of inputs so as to generate each output corresponds to input.

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT NO:12

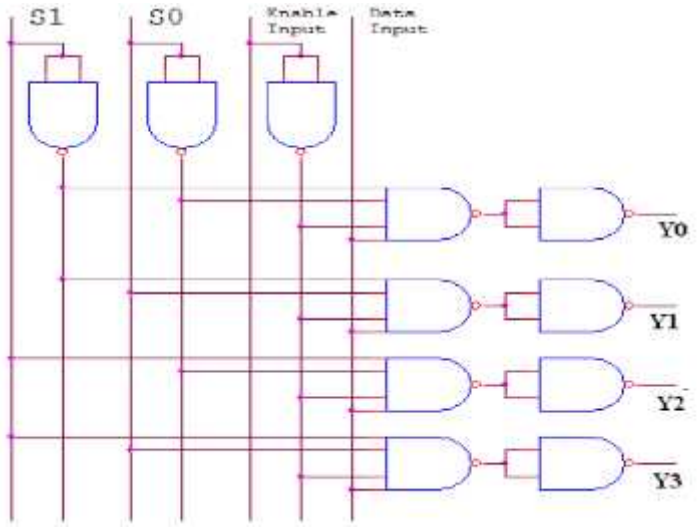
AIM: Design and implementation of DEMULTIPLEXER .

APPRATUS REQUIRED:

THEORY : De-multiplexers perform the opposite function of multiplexers. They transfer a small number of information units (usually one unit) over a larger number of channels under the control of selection signals. The general de-multiplexer circuit has 1 input signal, n control/select signals and 2n output signals. De-multiplexer circuit can also be realized using a decoder circuit with enable.

CIRCUIT DIAGRAM:

DE-MUX USING NAND GATES:



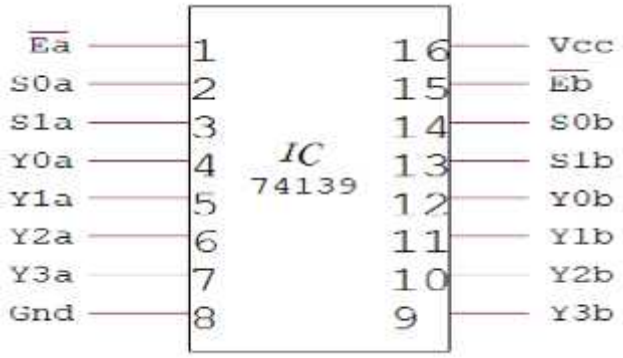
TRUTH TABLE:

Enable input	Data input	Select inputs		Outputs			
E	D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
1	0	X	X	X	X	X	X
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	1	0	0
0	1	1	1	1	0	0	0

IC 74139 (DEMUX) AND TRUTH TABLE:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual



Inputs			Outputs			
Ea	S1	S0	Y_3	Y_2	Y_1	Y_0
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

PROCEDURE:

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

CONCLUSION:

A **demultiplexer** (or demux) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. A multiplexer is often used with a complementary **demultiplexer** on the receiving end.

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

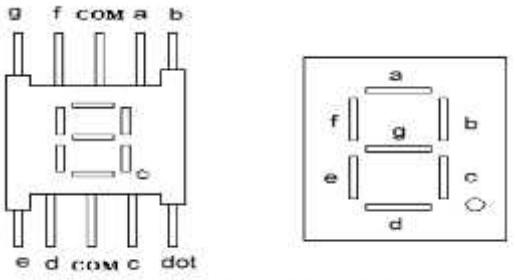
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT NO-13

AIM: Design and implementation of BCD TO SEVEN SEGMENT DECODER.

APPRATUS REQUIRED: IC7447, 7-Segment display (common anode), Patch chords, resistor (1K_) & IC Trainer Kit .

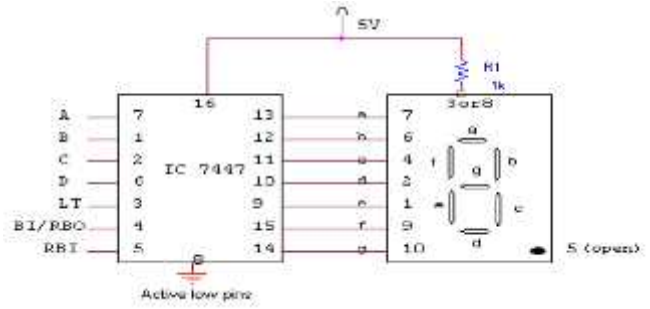
THEORY: The Light Emitting Diode (LED) finds its place in many applications in these modern electronic fields. One of them is the Seven Segment Display. Seven-segment displays contains the arrangement of the LEDs in "Eight" (8) passion, and a Dot (.) with a common electrode, lead (Anode or Cathode). The purpose of arranging it in that passion is that we can make any number out of that by switching ON and OFF the particular LED's. Here is the block diagram of the Seven Segment LED arrangement. The Light Emitting Diode (LED), finds its place in many applications in this modern electronic fields. One of them is the Seven Segment Display. Seven-segment displays contains the arrangement of the LEDs in "Eight" (8) passion, and a Dot (.) with a common electrode, lead (Anode or Cathode). The purpose of arranging it in that passion is that we can make any number out of that by switching ON and OFF the particular LED's. Here is the block diagram of the Seven Segment LED arrangement.



Seven-Segment Display

LED's are basically of two types-
Common Cathode (CC) -All the 8 anode legs uses only one cathode, which is common. Common Anode (CA)-The common leg for all the cathode is of Anode type. A decoder is a combinational circuit that connects the binary information from 'n' input lines to a maximum of 2n unique output lines. The IC7447 is a BCD to 7-segment pattern converter. The IC7447 takes the Binary Coded Decimal (BCD) as the input and outputs the relevant 7 segment code.

CIRCUIT DIAGRAM:



TUTH TABLE:

Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

CONCLUSION:

A decoder is a combinational circuit which is used to convert a binary or **BCD** (Binary Coded Decimal) number to the corresponding decimal number . It can be a simple binary to decimal decoder or a **BCD to 7 segment** decoder. Another relevant section is the combinational logic circuitry

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

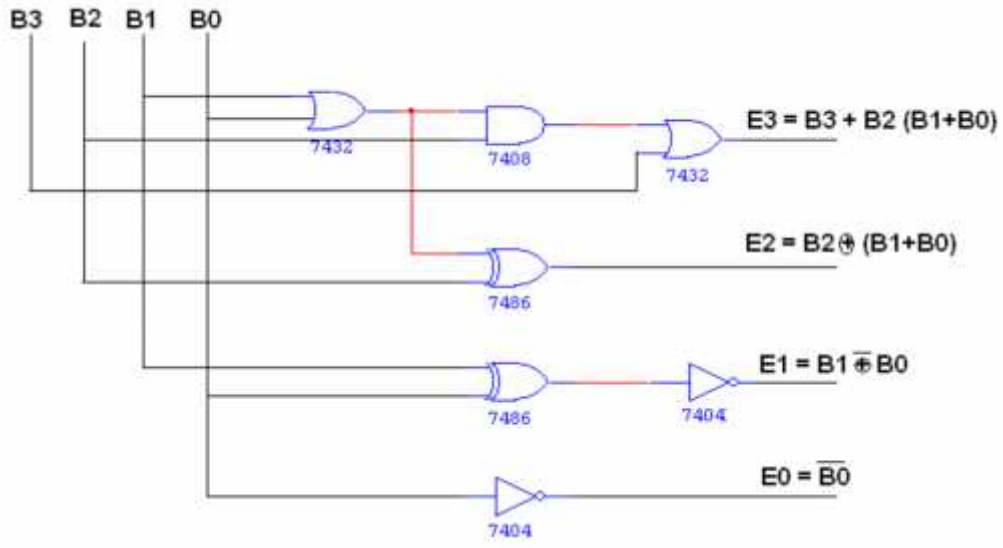
EXPERIMENT NO:14

AIM: Design and implementation of BCD TO EXCESS 3CODE CONVERTOR..

APPRATUS REQUIRED: IC 7486, IC 7432, IC 7408, IC 7400, etc.

THEORY: Code converter is a combinational circuit that translates the input code word into a new corresponding word. The excess-3 code digit is obtained by adding three to the corresponding BCD digit. To Construct a BCD-to-excess-3-code converter with a 4-bit adder feed BCDcode to the 4-bit adder as the first operand and then feed constant 3 as the second operand.
The output is the corresponding excess-3 code.
To make it work as a excess-3 to BCD converter, we feed excess-3 code as the first operand and then feed 2's complement of 3 as the second operand. The output is the BCD code.

CIRCUIT DIAGRAM:



TRUTH TABLE:

BCD				EXCESS 3			
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

PROCEDURE:

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs.

CONCLUSION:

The Excess-3 BCD system is formed by adding 0011 to each BCD value THE *BCD TO EXCESS 3* CODE CONVERTER

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT NO:15

AIM: To design and implement SR latch, SR flip flop and JK flip flop.

APPRATUS REQUIRED: IC 7486, IC 7432, IC 7408, IC 7400, etc.

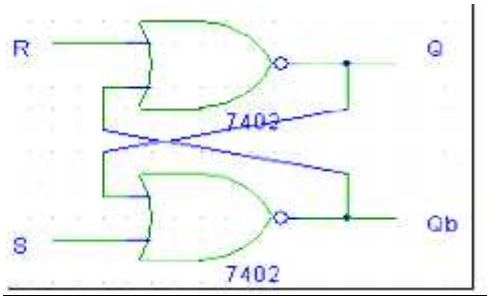
THEORY: Logic circuits that incorporate memory cells are called *sequential logic circuits*; their output depends not only upon the present value of the input but also upon the previous values. Sequential logic circuits often require a timing generator (a clock) for their operation. The latch (flip-flop) is a basic bi-stable memory element widely used in sequential logic circuits. Usually there are two outputs, Q and its complementary value. Some of the most widely used latches are listed below.

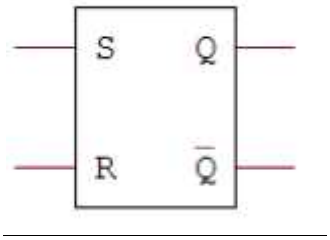
SR LATCH:

An S-R latch consists of two cross-coupled NOR gates. An S-R flip-flop can also be design using cross-coupled NAND gates as shown. The truth tables of the circuits are shown below. A clocked S-R flip-flop has an additional clock input so that the S and R inputs are active only when the clock is high. When the clock goes low, the state of flip-flop is latched and cannot change until the clock goes high again. Therefore, the clocked S-R flip-flop is also called "enabled" S-R flip-flop.

CIRCUIT DIAGRAM:

R-S LATCH:





TRUTH TABLE:

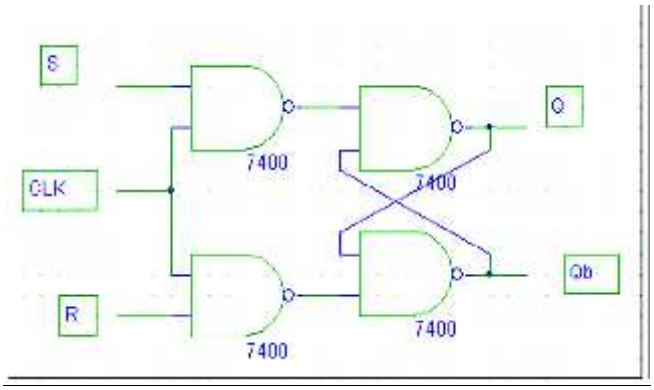
S	R	Q	Q'
0	0	NC	NC

0	1	0	1
1	0	1	0
1	1	FB	FB

TRUTH TABLE:

S	R	Q	Q’
0	0	FB	FB
0	1	1	0
1	0	0	1
1	1	NC	NC

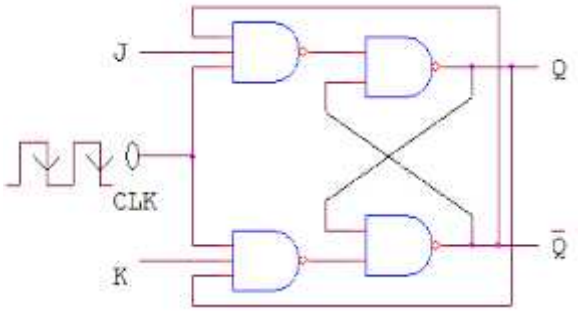
S-R FLIP FLOP:



TRUTH TABLE:

CLK	S	R	Q	Q’
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

J-K FLIP FLOP :
CIRCUIT DIAGRAM AND TRUTH TABLE:



PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

CONCLUSION:

n **electronics**, a **flip-flop** or latch is a circuit that has two stable states and can be used to store state information. A **flip-flop** is a bistable multivibrator. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs.

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

EXPERIMENT NO:16

AIM: To design and implement ALU.

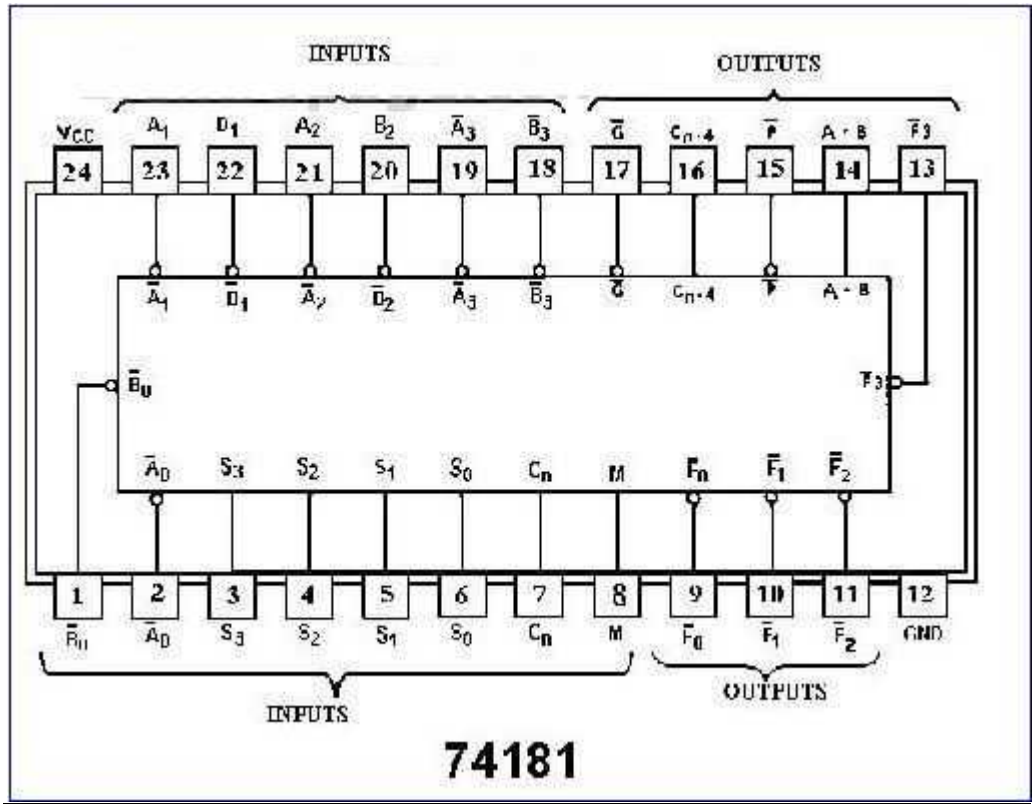
APPRATUS REQUIRED: IC 74181

THEORY: ALU (AIRTHMATIC LOGIC UNIT) is a circuit which performs arithmetic and logical operations. Arithmetic operations are like addition ,substraction, multiplication, and division. Logical operations are like and, or nand, nor ,not operations on bits. Here we will design the ALU for addition, substration and all logical operations. For this we need to design circuits for all the arithmetic and logical operations we want to perform. All these circuits then will be multiplexed through multiplexers. keeping in mind the complexity of circuit we will multiplexer only some number of circuits. for a particular operation we have to select that particular circuit through multiplexer by selecting appropriate select lines of multiplexer.

CIRCUIT DIAGRAM:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual



PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

CONCLUSION:

The 74181 is a bit slice arithmetic logic unit (ALU), implemented as a 7400 series TTL integrated circuit. The first complete ALU on a single chip,[1] it was used as the arithmetic/logic core in the CPUs of many historically significant minicomputers and other devices.

The 74181 represents an evolutionary step between the CPUs of the 1960s, which were constructed using discrete logic gates, and today's single-chip CPUs or microprocessors.

PRECAUTIONS:

1. Digital IC trainer kit must be switched off while connecting the wires.
2. IC Chips must be handled carefully so that pins could not be damaged.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual