UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: UNIX & Shell Programming **Subject Code-MCA301**
Year: 2ND Year **Semester: SECOND**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**5L**
	1. Organization of UNIX.	2L
	2. User interface, Programmer interface.	3L
2	**System calls**	**12L**
	1. The environment of UNIX process System calls.	4L
	2. Process control, File related system calls.	4L
	3. Process related system calls. Signals programming using system calls.	4L
3	**I/O**	**4L**
	1. Advanced I/O multiplexing. Memory mapped I/O.	4L
4	**Inter-process communications**	**9L**
	1. Interprocess communication: Pipes, shared memory, semaphores, messages.	5L
	2. Advanced inter-process communications. Streams, Pipes, Open server	4L

Faculty In-Charge HOD, CSE Dept.

Assignment:

Module-1 (Introduction):

1. User interface vs programmer interface

Module-2 (System calls):

1. Different system calls.

Module-3 (I/O):

1. Advanced Memory mapped I/O

Module-4 (Interprocess communication):

1. Interprocess communication
2. Pipes

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Cloud Computing **Code-MCA302**
Year: 2nd Year **Semester: Third**

Module Number	Topics	Number of Lectures
1	**Cloud Computing Fundamental, Business Agility**	**6L**
	Cloud computing definition,private,public and hybrid cloud. Cloud types, Iaas, Saas. Benefits and challenges of cloud computing. Role of virtualization in enabling the cloud.	
2	**Cloud Applications**	**5L**
	Technologies and the processes required when deploying web services; Deploying a web service from inside and outside a cloud architecture, advantages and disadvantages	
3	**Cloud Services Management, Cloud Economics**	**9L**
	Reliability, availability and security of services deployed from the cloud. Cloud Computing infrastructures available for implementing cloud based services. Economics of choosing a Cloud platform for an organization, based on application requirements, economic constraints and business needs.	
4	**Application Development**	**8L**
	Service creation environments to develop cloud based applications. Development environments for service development; Amazon, Azure, Google App.	
5	**Best Practice Cloud IT Model**	**7L**
	Analysis of Case Studies when deciding to adopt cloud computing architecture. How to decide if the cloud is right for your requirements. Cloud based service, applications and development platform deployment so as to improve the total cost of ownership (TCO)	
Total Number Of Hours = 35		

Faculty In-Charge HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Cloud Computing Code-CS701
Year: 4th Year Semester: First

Assignment:

Module-1 (Cloud Computing Fundamental, Business Agility):

1. Explain private,public and hybrid cloud. What are the different types of Cloud are present?
2. Mention the Benefits and challenges of cloud computing.

Module-2 (Cloud Applications):

1. What are the technologies and the processes required when deploying web services in cloud?
2. Explain the deployment of web service from inside and outside a cloud architecture.

Module-3 (Cloud Services Management, Cloud Economics):

1. Explain reliability, availability and security of services deployed from the cloud.
2. What are the economic constraints and business needs to implement cloud environment?

Module-4 (Application Development):

1. How service creation environments are used to develop cloud based applications?
2. Explain how development environments for service development are used.

Module-5 (Best Practice Cloud IT Model):

1. How do you decide if the cloud is right for your requirements?
2. Explain Cloud based service.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Title of Course: Intelligent System
Course Code: MCA303
L-T: 3-1 **Course Credits: 4**

Objective: Introducing concepts, models, algorithms, and tools for development of intelligent systems. Example topics include artificial neural networks, genetic algorithms, fuzzy systems, swarm intelligence, ant colony optimization, artificial life, and hybridizations of the above techniques. Students will be able to sense these techniques from a machine learning perspective. This domain is called Computational Intelligence, and is a numerical interpretation of biological intelligence.

Learning Outcome: On the completion of this course, the student will have:

- An understanding of fundamental computational intelligence and machine learning models.
- Implemented neural networks, genetic algorithms, and other computational intelligence and machine learning algorithms.
- Applied computational intelligence and machine learning techniques to classification, prediction, pattern recognition, and optimization problems.

Course Contents:

Computational intelligences, agents, example application domains, Representation and reasoning systems, Datalog, syntax and semantics, variables, queries, answers, recursion. Proofs, soundness, completeness, top-down and bottom-up reasoning, function symbols, Searching, graphics, generic search engine, blind search strategies, heuristic search, A* search. Pruning the search space, search direction, iterative deepening, dynamic programming, constraint satisfaction, consistency algorithms, hill climbing, randomized algorithms. Knowledge representation issues, defining a solution, choosing a representation, semantic networks, frames, primitive and derived relations. Equality, inequality, unique names assumption, complete knowledge assumption, negation as failure. Actions and planning. STRIPS representation, situation calculus, forward planning, resolution and planning. The STRIPS planner, Midterm, Regression Planning. A building situated robots Robot Architectures

Textbooks:

1. Computational Intelligence : Concepts to Implementations by Eberhart & Shi

Reference Books:

1. Introduction to Genetic Algorithms by Melanie Mitchell
2. Handbook of Genetic Algorithms by Davis
3. Machine Learning by Tom Mitchell

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Operating System and System Soft wares — Subject Code-MCA304
Year: 2nd Year — Semester: Third

Module Number	Topics	Number of Lectures
1	**Introduction:**	**4L**
	1. Introduction to OS. Operating system functions, evaluation of O.S	1
	2. Evaluation of O.S., Different types of O.S.	1
	3. batch, multi-programmed, time-sharing	1
	4. Different types of O.S.: real-time, distributed, parallel	1
2	**System Structure:**	**3L**
	1. Computer system operation, I/O structure, storage structure	2
	2. O/S services, system calls.	1
	Process Management:	
4.	**Processes:**	**3L**
	1. Concept of processes, process scheduling, operations on processes	3
	CPU scheduling:	**5L**
	1. scheduling criteria, preemptive & non-preemptive scheduling	2
	2. scheduling algorithms (FCFS, SJF, RR, priority),	3
6	**Process Synchronization:**	**4L**
	1. background, critical section problem, critical region	2
	2. Synchronization hardware, classical problems of synchronization, semaphores.	2
7	**Deadlocks:**	**5L**
	1. System model, deadlock characterization	1
	2. Methods for handling deadlocks, deadlock prevention	2
	3. Deadlock avoidance, deadlock detection	2
9	**Virtual Memory:**	**3L**
	1. Demand paging, performance	1
	2. Page replacement, page replacement algorithms (FCFS, LRU)	2
12	**Disk Management:**	**3L**
	1. disk structure, disk scheduling (FCFS, SSTF, SCAN,C-SCAN) ,disk reliability,	3

13	**Protection & Security:**	
	Protection & Security:	**2L**
	1. Goals of protection, domain of protection, Security problem, authentication	2
	2. Worm and Viruses	1
Total Number Of Hours = 32		

Faculty In-Charge HOD, CSE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Assignment:

1. What is the job of operating system?
2. Explain the benefits of Multithreading?
3. Describe the process life cycle with proper diagram
4. Describe different type of Disk Scheduling.
5. What is fragmentation? Why we need fragmentation?
6. Describe the necessary conditions for Deadlock.
7. What do you mean by starvation explain with proper example?
8. Using " preemptive shortest remaining time first" find out the Average waiting time.

Process Name	Burst Time	Arrival Time
P_1	3	0
P_2	6	2
P_3	4	4
P_4	5	6
P_5	2	8

9. Difference between Anti-Virus and Firewall.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Management Accounting** Subject Code: **HU301**
Year: **2nd Year** Semester: **THIRD**

Module Number	Topics	Number of Lectures
1	**Background** - Nature of Management Accounting **Financial Analysis** - Cash Flow Statement (as per AS3), Financial Statements Analysis	**7**
2	**Cost Accumulation** - Fundamentals of Job-Order Batch & Process Costing, Variable Costing and Absorption (Full) Costing, Activity Based Costing System	9
3	**Profit Planning** - Cost -Volume-Profit Analysis, Budgeting and Profit Planning, Flexible Budgeting	8
4	**Cost Control** - Standard Costs and quality Costs, Cost Variance Analysis, Revenue and Profit Variance Analysis, Responsibility Accounting **Relevant Costing –** Introduction – Relevant Costs and Revenues- Cost Concepts – Outsourcing Decision – Decision to accept or reject a special order – Decision to continue or abandon a project	**9**
5	**Total Cost Management** – Introduction – TCM and Business competitive edge - TCM Principles and implementation	**8**
Total Number Of Hours = 41 L		

Prof DR Preeti Sharma **Prof DR Preeti Sharma**

Faculty In-Charge HOD, Management Dept

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Statistics & Numerical Techniques Subject Code-M301
Year: 2nd Year Semester: THIRD

Module Number	Topics	Number of Lectures (25)
	Errors	**2**
	Approximationin numericalcomputation,Truncationandrounding errors	
	Interpolation	**5**
	Lagrange's interpolation,Newtonforwardand backward differences interpolation,Newtondivided difference.	
	NumericalIntegration	**3**
	Trapezoidal rule,Simpson 1/3 rule,Weddle'srule	
	Numericalsolutionofasystemof linear equation	5
	Gausseliminationmethod, Matrix inversion,LU factorizationmethod,Gauss-Jacobimethod,GaussSeidelmethod.	
	AlgebraicEquation	**5**
	Bisection method,Secantmethod,Regula–Falsimethod,NewtonRaphsonmethod,Method ofIteration	
	Numericalsolutionofordinarydifferentialequation	**5**
	Taylor'sseriesmethod,Euler'smethod,Runga–kuttamethod,predictor–correction method	

Faculty In-Charge HOD, Maths Dept.

Assignment:

Errors:

1. Find the relative error if 2/3 is approximated to 0.667.
2. Find the percentage error if 625.483 is approximated to three significant figures.
3. Find the relative error in taking $\pi = 3.141593$ as 22/7.
4. The height of an observation tower was estimated to be 47 m, whereas its actual height was 45 m. calculate the percentage relative error in the measurement.
5. Two numbers are 3.5 and 47.279 both of which are correct to the significant figures given. Find their product.

Interpolations:

1. Apply Newton's backward Interpolation to the data below, to obtain a polynomial of degree4 in x

x:	1	2	3	4	5
$f(x)$:	1	-1	1	-1	1

2. Using Newton's backward Interpolation, find the value of $f(2)$ from the following table:

x:	1	3	4	5	6	7
$f(x)$:	2.68	3.04	3.38	3.68	3.96	4.21

3. Using Newton's Forward Interpolation, the area A of a circle of diameter d.

d:	80	85	90	95	100
A:	5026	5674	6362	7088	7854

Calculate the area of a circle of diameter 105.

4. Estimate the value of $f(22)$ and $f(42)$ from the following available data:

x:	20	25	30	35	40	45
$f(x)$:	354	332	291	260	231	204

Using Newton's Forward Interpolation

5. Find $f(x)$ as a polynomial in x for the following data by Newton's divided difference method:

x:	-4	-1	0	2	5
$f(x)$:	1245	33	5	9	1335

6. Using Newton's divided difference method to find $f(x)$ from the following available data:

x:	0	1	2	4	5	6
$f(x)$:	1	14	15	5	6	19.

Numerical Integrations:

1. Apply trapezoidal rule to find the integral $I = \int_0^1 \sin \pi x \, dx.$
2. Find, from the following table the area bounded by the curve and the x-axis from $x = 7.47$ to $x=7.52$, $f(7.47) = 1.93, f(7.48) = 1.95, f(7.49) = 1.98, f(7.50) = 2.01,$ $f(7.51) = 2.03, f(7.52) = 2.06.$
3. Evaluate $I = \int_0^1 \frac{1}{1+x^2} dx$, correct to three decimal places and also find the approximate value of π.
4. A solid of revolution is formed by rotating about the x-axis the area between the x-axis, the lines $x = 0$ and $x = 1$ and a curve through the points with the following coordinates: $(0,1),(0.25,0.9896)(0.5,0.9589),(0.75,0.9089),(1,0.8415).$

AlgebraicEquation:

1. Find the root of the following equations correct three decimal places by the Regulafalsi method: $x^3 + x - 1 = 0$.
2. Using Regulafalsi method, compute the real root of the following equation correct to four decimal places: $xe^x = 2$.
3. Find the root of the following equations correct three decimal places by the Regulafalsi method: $x^6 - x^4 - x^3 - 1 = 0$.
4. Find the root of the following equations correct three decimal places by the bisection method : $x - e^x = 0$

5. Find the root of the following equations, using the bisection method correct three decimal places: $x - \cos x = 0$
6. Using the bisection method to find a root of the equation to four decimal places: $x^3 - 9x + 1 = 0$

Numericalsolutionofordinarydifferentialequation:

1. Using Runge-kutta method of order 4, find $y(0.2)$ given that $\frac{dy}{dx} = 3x + \frac{1}{2}y, y(0) = 1$ taking $h = 0.1$.
2. Using Runge-kutta method of order 4, compute $y(0.2)$ and $y(0.4)$ from $10\frac{dy}{dx} = x^2 + y^2, y(0) = 1$ taking $h = 0.1$.
3. Using Milne's predictor-corrector method to obtain the solution of the equation $\frac{dy}{dx} = x - y^2$ at $x = 0.8$ given that $y(0) = 0.0000$, $y(2) = 0.0200$, $y(4) = 0.0795$ $y(6) = 0.1762$.
4. Given $2\frac{dy}{dx} = (1 + x^2)y^2$ and $y(0) = 1$, $y(0.1) = 1.06$, $y(0.2) = 1.12$, $y(0.3) = 1.21$, evaluate $y(0.4)$ by Milne's predictor-corrector method.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Unix LAB
Course Code: MCA391
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:
This course introduces basic understanding of UNIX OS, UNIX commands and File system and to familiarize students with the Linux environment. To make student learn fundamentals of shell scripting and shell programming. Emphases are on making student familiar with UNIX environment and issues related to it..

Learning Outcomes:

Upon completion of this course, the student will be able to:

1. You will be able to run various UNIX commands on a standard UNIX/LINUX Operating system (We will be using Ubuntu flavor of the Linux operating system).
2. You will be able to run C / C++ programs on UNIX.
3. You will be able to do shell programming on UNIX OS.
4. You will be able to understand and handle UNIX system calls.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1: Installation of Unix/Linux operating system.
Exercise No. 2: Write a C program to emulate the UNIX ls-l command.
Exercise No. 3: Write a C program to check the given integer is prime or not.
Exercise No. 4: Write a C program to display Largest of three numbers.
Exercise No. 5: Write a shell script program to display list of user currently logged in.
Exercise No. 6: Write a shell script program to display HELLO WORLD
Exercise No. 7: Write a shell script program to develop a scientific calculator
Exercise No. 8: Write a grep/egrep script to find the number of words character, words and lines in a file.
Exercise No. 9: Shell programming.
Exercise No. 10: Write a shell script program to display the process attributes.
Exercise No. 11: Write a shell script program to check variable attributes of file and processes.
Exercise No. 12: Installation of VirtualBox (VMWare) on a PC having other operating system.
Exercise No. 13: Shell Script program for changing process priority.

Text Book:
1. Maurice J. Bach, Design of the UNIX Operating System, PHI.

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.

Experiment No: 2

AIM:

Write a C program to emulate the Unix ls-l command.

Program:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
int main()
{
int pid;            //process id
pid = fork();     //create another process
if ( pid < 0 )
{                                  //fail
printf("\nFork failed\n");
exit (-1);
}
else if ( pid == 0 )
{                     //child
execlp ( "/bin/ls", "ls", "-l", NULL );  //execute ls
}
else
{                             //parent
wait (NULL);              //wait for child
printf("\nchild complete\n");
exit (0);
}
}
```

Output:

guest-glcbIs@ubuntu:~$gcc –o lsc.out lsc.c

guest-glcbIs@ubuntu:~$./lsc.out

total 100

-rwxrwx—x 1 guest-glcbls guest-glcbls 140 2012-07-06 14:55 f1

drwxrwxr-x 4 guest-glcbls guest-glcbls 140 2012-07-06 14:40 dir1

child complete

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Statistics & Numerical Analysis Lab
Course Code: M392
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:
- To give an overview of *what* can be done
- To give insight into *how* it can be done
- To give the confidence to tackle numerical solutions

An understanding of how a method works aids in choosing a method. It can also provide an indication of what can and will go wrong, and of the accuracy which may be obtained.
- To gain insight into the underlying physics

The aim of this course is to introduce numerical techniques that can be used on computers, rather than to provide a detailed treatment of accuracy or stability.

Learning Outcomes:
On completion of this course, the student will be able to:
1. Demonstrate skills in using computer programming tools for engineering calculations;
2. Demonstrate ability to construct simple computer algorithms using a programming tool;
3. Apply simple numerical methods to solve mathematical problems with relevance to civil engineering;
4. Appreciate the limitations and the applicability of the numerical methods;
5. Apply computer-based numerical methods for the solution of engineering problems.

Course Contents:
Exercises that must be done in this course are listed below:
1. Assignments on Newton forward /backward, Lagrange' s interpolation.
2. Assignments on numerical integration using Trapezoidal rule, Simpson' s 1/3 rule.
3. Assignments on numerical solution of a system of linear equations using Gauss elimination and Gauss-Seidel iterations.
4. Assignments on numerical solution of Algebraic Equation by Regular-falsi and Newton Raphson methods.
5. Assignments on ordinary differential equation: Euler' s and Runga-Kutta methods.

Text Book:
- Introductory method of numerical analysis, Sastry S.S
- Computer Programming in fortran 77, Rajaraman V
- Numerical methods: for scientific and engineering computation,Mahinder Kumar Jain

.

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.

Experiment No: 1(a) Newton forward interpolation
Aim: Write a C program to implement the Newton forward interpolation.
Description:
Interpolation is the process of finding the values of y corresponding to the any value of x between x0 and xn for the given values of y=f(x) for a set of values of x. Out of the many techniques of interpolation, Newton's Forward and Backward Interpolation are two very widely used formulas. In this tutorial, we're going to discuss a C program for Newton Forward Interpolation along with its sample output.
Both of Newton's formulas are based on finite difference calculus. These formulas are very often used in engineering and related science fields. Before going through the source code for Newton Forward Interpolation, let's go through the forward interpolation formula and the variables used in the C program.
Newton's forward interpolation formula contains y0 and the forward differences of y0. This formula is used for interpolating the values of y near the beginning of a set of tabulated values and

$$P(x) = y_0 + q\,\Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_{-1} + \frac{(q+1)\,q(q-1)}{3!}\Delta^3 y_{-1} +$$
$$+ \frac{(q+1)\,q(q-1)(q-2)}{4!}\Delta^4 y_{-2} + \frac{(q+2)(q+1)\,q(q-1)(q-2)}{5!}\Delta^5 y_{-2} + \ldots$$
$$\ldots + \frac{(q+n-1)\ldots(q-n+1)}{(2n-1)!}\Delta^{2n-1} y_{-(n-1)} + \frac{(q+n-1)\ldots(q-n)}{(2n)!}\Delta^{2n} y_{-n},$$

Compared to forward interpolation, the backward interpolation formula contains yn and the backward differences of yn. This formula is used for interpolating the values of y near the end of a set of tabulated values and also for extrapolating the values of y a little ahead (i.e. to the right) of yn.

Algorithm:

1. Function NFI ()
2. Read n, x
3. For I = 1 to n by 1 do
4. Read x[i], y[i]
5. End for
6. If ((x < x[i] or (x > x[n]))
7. Print “Value lies out of boundary”
8. Exit
9. End if
10. //Calculating p
11. p = (x – x [1]) / (x [2]-x [1])
12. // Forward diff table
13. For j = 1 to (n-1) by 1 do
14. For i =1 to (n - j) by 1 do
15. If (j=1) Then
16. d[i][j] = y [i+1] – y[i]
17. Else
18. d[i][j] = d[i+1][j-1] – d[i][j-1]
19. End if
20. End For
21. End For
22. // Applying Formula
23. Sum =y [1]
24. For I = 1 to (n-1) by 1 do
25. Prod = 1
26. For j =0 to (i-1) by 1 do
27. Prod = prod * (p-j)
28. End for
29. m = fact(i)
30. Sum = sum + (d[1][i] * prod) / m
31. End For
32. Print “Ans is”, Sum
33. End Function

/* Program to implement Newton’s forward interpolation*/

```
1       #include<stdio.h>
2       #include<conio.h>
3       #include<math.h>
4       #include<stdlib.h>
5       main()
6       {
7         float x[20],y[20],f,s,h,d,p;
8         intj,i,n;
9         printf("enter the value of n :");
10        scanf("%d",&n);
11        printf("enter the elements of x:");
```

```
13      {
14          scanf("%f",&x[i]);
15           }
16                  printf("enter the elements of y:");
17               for(i=1;i<=n;i++)
18               {
19              scanf("%f",&y[i]);
20                          }
21     h=x[2]-x[1];
22     printf("Enter the value of f:");
23     scanf("%f",&f);
24    s=(f-x[1])/h;
25    p=1;
26    d=y[1];
27    for(i=1;i<=(n-1);i++)
28     {
29                  for(j=1;j<=(n-i);j++)
30                   {
31                        y[j]=y[j+1]-y[j];
32
33                   }
34                   p=p*(s-i+1)/i;
35                   d=d+p*y[1];
36     }
37    printf("For the value of x=%6.5f THe value is %6.5f",f,d);
38     getch();
39    }
```

OUTPUT:

```
how many record you will be enter: 5
enter the value of x0: 2.5
enter the value of f(x0): 9.75
enter the value of x1: 3
enter the value of f(x1): 12.45
enter the value of x2: 3.5
enter the value of f(x2): 15.70
enter the value of x3: 4
enter the value of f(x3): 19.52
enter the value of x4: 4.5
enter the value of f(x4): 23.75
Enter X for finding f(x): 4.25
u = -0.500
f(4.25) = 21.583750
```

Experiment No: 1(b) Newton backward interpolation
Aim: Write a C program to implement Newton backward interpolation.
Algorithm:

1. Function NBI ()
2. Read n, x
3. For I = 1 to n by 1 do
4. Read x[i], y[i]
5. End for
6. If ((x < x[i] or (x > x[n]))
7. Print “Value lies out of boundary”
8. Exit

10. //Calculating p
11. p = (x – x [1]) / (x [2]-x [1])
12. // Forward diff table
13. For j = 1 to (n-1) by 1 do
14. For i =1 to (n - j) by 1 do
15. If (j=1) Then
16. d[i][j] = y [i+1] – y[i]
17. Else
18. d[i][j] = d[i+1][j-1] – d[i][j-1]
19. End if
20. End For
21. End For
22. // Applying Formula
23. Sum =y [n]
24. For I = 1 to (n-1) by 1 do
25. Prod = 1
26. For j =0 to (i-1) by 1 do
27. Prod = prod * (p+j)
28. End for
29. m = fact(i)
30. Sum = sum + (d[n-1][i] * prod) / m
31. End For
32. Print “Ans is”, Sum
33. End Function

/* Program to implement Newton’s forward interpolation */

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
main()
{
float x[20],y[20],f,s,d,h,p;
intj,i,k,n;
printf(“enter the value of the elements :”);
scanf(“%d”,&n);
printf(“enter the value of x: \n\n”);
for(i=1;i<=n;i++)
    {
scanf(“%f”,&x[i]);
                }
printf(“enter the value of y: \n\n”);
for(i=1;i<=n;i++)
    {
scanf(“%f”,&y[i]);
                }
                h=x[2]-x[1];
printf(“enter the searching point f:”);
scanf(“%f”,&f);
s=(f-x[n])/h;
d=y[n];
p=1;
for(i=n,k=1;i>=1,k<n;i–,k++)
{
for(j=n;j>=1;j–)
            {
y[j]=y[j]-y[j-1];
```

```
                    p=p*(s+k-1)/k;
                    d=d+p*y[n];
}
printf("for f=%f ,ans is=%f",f,d);
getch();
}
```

OUT PUT:

```
how many record you will be enter: 5
enter the value of x0: 2.5
enter the value of f(x0): 9.75
enter the value of x1: 3
enter the value of f(x1): 12.45
enter the value of x2: 3.5
enter the value of f(x2): 15.70
enter the value of x3: 4
enter the value of f(x3): 19.52
enter the value of x4: 4.5
enter the value of f(x4): 23.75
Enter X for finding f(x): 4.25
```

x(i)	y(i)	y1(i)	y2(i)	y3(i)	y4(i)
2.500	9.750				
3.000	12.450	2.700			
3.500	15.700	3.250	0.550		
4.000	19.520	3.820	0.570	0.020	
4.500	23.750	4.230	0.410	-0.160	-0.180

```
 u = -0.500
f(4.25) = 21.583750 -
```

Experiment No: 1(c)Lagrange' s interpolation
Aim: Write a C program to implement Lagrange' s interpolation.

Algorithm:

1. Input number of Observation n
2. For i = 1 to n
3. Input Xi
4. Input Yi
5. Next i
6. Input xp at which yp to be computed
7. Initialize yp = 0
8. For i = 1 to n
9. t = 1
10. For j = 1 to n
11. If $j \neq i$
12. t = t * (xp - Xj)/(Xi - Xj)
13. End If
14. Next j

16. Next i
17. Print yp as output
18. Stop

Program to implement Lagrange' s interpolation*/

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
 float x[10],y[10],temp=1,f[10],sum,p;
 inti,n,j,k=0,c;

 printf("\nhow many record you will be enter: ");
 scanf("%d",&n);
 for(i=0; i<n; i++)
 {
  printf("\n\nenter the value of x%d: ",i);
  scanf("%f",&x[i]);
  printf("\n\nenter the value of f(x%d): ",i);
  scanf("%f",&y[i]);
 }
 printf("\n\nEnter X for finding f(x): ");
 scanf("%f",&p);

 for(i=0;i<n;i++)
 {
   temp = 1;
   k = i;
   for(j=0;j<n;j++)
   {
     if(k==j)
     {
       continue;
     }
     else
     {
       temp = temp * ((p-x[j])/(x[k]-x[j]));
     }
   }
   f[i]=y[i]*temp;
 }

 for(i=0;i<n;i++)
 {
    sum = sum + f[i];
 }
 printf("\n\n f(%.1f) = %f ",p,sum);
 getch();
}
```

OUTPUT:

```
enter the value of n 4
enter the value to be found 2.5
enter the values for xi's &fi's
1 1
2 8
```

4 64
X = 2.500000
sum = 15.625000

Experiment No:2.Trapezoidal rule
Aim: Write **a C program to implement Trapezoidal rule**.
Description:
A number of definite integrals need to be solved in applied mathematics, physics and engineering. The manual analytical solution of definite integrals is quite cumbersome and time consuming. So, in this post I have presented source code in C program for Trapezoidal method as one of the computer-programming-based solutions of definite integrals. The techniques of numerical methods are used to solve this equation; it involves a number of calculations and efforts have been made to minimize error in the program.
The trapezium or trapezoidal rule can be used as a way of estimating the area under a curve because the area under a curve is given by integration. So, the trapezoidal rule gives a method of estimating integrals. This is useful when you come across integrals that you don't know how to evaluate. So, the program for trapezoidal method in C given here is applicable to calculate finite integral or area under a curve.
$h=(x_n - x_0)/n$
After that, the C source code for trapezoidal method uses the following formula to calculate the value of definite integral:

$$\int_{x_0}^{x_n} f(x)\,dx = \frac{1}{2}h[(y_0 + y_n) + 2(y_1 + y_2 + \ldots + y_{n-1})]$$

Algorithm:

1. Read x1, x2, e {x1 and x2 are the two end points of the internal the allowed error in integral is e}
2. h = x2 - x1
3. SI = (f(x1) + f(x2))/2;
4. I = h - si
5. i = 1 Repeat
6. x = x1 + h/2
7. for J= 1 to I do
8. SI = SI + f(x)
9. x = x + h
10. End for
11. i = 21
12. h = h/2 {Note that the internal has been halved above and the number of points where the function has to be computed is doubled}
13. i0 = i1
14. i1 = h.si. until / I1 - i0 / <= c./i1/
15. Write I1, h, i
16. Stop.

/* Program to to implement Trapezoidal rule */

```
#include<stdio.h>
#include<math.h>
main()
{
float h, a, b, n, x[20], y[20], sum = 0, integral;
int i;
clrscr();
printf("enter the value of a, b, n:");
scanf("%f %f %f", &a, &b, &n);
printf("enter the values of x:");
```

```
 {
scanf("%f", &x[i]);
 }
printf("\n enter the values of y:");
for(i = 0; i <= (n-1); i++)
 {
scanf("%f", &y[i]);
 }
 h = (b-a)/n;
x[0] = a;
for(i = 1; i <= n-1; i++)
 {
x[i] = x[i-1] + h;
sum = sum + 2 * y[i];
 }
sum = sum + y[b];
integral = sum * (h/2);
printf("approximate integral value is: %f", integral);
getch();
}
```

OUTPUT :

```
enter the values of a, b, n
123
enter the values of x:
123
enter the values of y:
123
approximate integral value is 2.166667
```

Experiment No:2(a)Simpson' s 1/3 rule

AIM: Write a C Program to implement Simpson' s 1/3 rule.

Description:

n the source code below, a function *f(x) = 1/(1+x)* has been defined. The calculation using **Simpson 1/3 rule in C** is based on the fact that the small portion between any two points is a parabola. The program follows the following steps for calculation of the integral.

- As the program gets executed, first of all it asks for the value of lower boundary value of x i.e. x_0, upper boundary value of x i.e. x_n and width of the strip, h.
- Then the program finds the value of number of strip as $n=(x_n - x_0)/h$ and checks whether it is even or odd. If the value of 'n' is odd, the program refines the value of 'h' so that the value of 'n' comes to be even.
- After that, this C program calculates value of f(x) i.e 'y' at different intermediate values of 'x' and displays values of all intermediate values of 'y'.
- After the calculation of values of 'c', the program uses the following formula to calculate the value of integral in loop.
  Integral = $*((y_0 + y_n) + 4(y_1 + y_3 + \ldots\ldots\ldots + y_{n-1}) + 2(y_2 + y_4 + \ldots\ldots\ldots + y_{n-2}))$
- Finally, it prints the values of integral which is stored as 'ans' in the program.

If f(x) represents the length, the value of integral will be area, and if f(x) is area, the output of Simpson 1/3 rule C program will be volume. Hence, numerical integration can be carried out using the program below; it is very easy to use, simple to understand, and gives reliable and accurate results.

f(x) = 1/(1+x)

.

Algorithm:

1. Read x1, x2, e
2. h = (x2 - x1)/2
3. i = 2

5. s2 = 0
6. s4 = f(x1 + h)
7. I0 = 0
8. In = (s + 4s4).(h/3)
9. Repeat
10. s2 = s2 + s4 { s2 stores already computed functional value and s4 the value computed in the new nitration }
11. s4 = 0
12. x = x1 + h/2
13. for j = 1 to I do
14. s4 = s4 + f(x)
15. x = x + h
16. h = h/2
17. i = 2i
18. io = in
19. in = (s1 + 2s2 + 4s4) . (h/3)
20. until |In-Io|≤e. /in
21. Write In, h, i
22. STOP

/* Program to to implement Simpson' s 1/3 rule. */

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
float h, a, b, n, x[20], y[20], sum = 0, itgl;
int i;
clrscr();
printf("enter the values of a, b, n");
scanf("%f%f%f", &a, &b, &n);
printf("enter the values of x");
for(i = 0; i <= n; i++)
 {
scanf("%f", &x[i]);
 }
printf("\n enter the values of y");
for(i = 0; i <= n; i++)
 {
scanf("%f", &y[i]);
 }
 h = (b - a)/n;
 a = x[0];
 b = x[n];
for(i = 0; i <= (n-2); i++)
 {
x[i] = x[i] + h;
if(i % 2 == 0)
  {
sum = sum + 4 * y[i];
  }
else
  {
sum = sum + 2 * y[i];
  }
 }
```

```
printf("integral value%f", itgl);
getch();
}
```

OUTPUT :

```
enter the values of a, b, n
123
enter the value of x
4567
enter the values of y
8912
integral value is 5.555556
```

Experiment No: 3(a)Gauss elimination.

AIM: Write a C Program to implement Gauss elimination method.

Description:

let us first consider the following three equations:

a1x + b1y + c1z = d1
a2x + b2y + c2z = d2
a3x + b3y + c3z = d3

Assuming a1 ≠ 0, x is eliminated from the second equation by subtracting (a2/ a1) times the first equation from the second equation. In the same way, the C code presented here eliminates x from third equation by subtracting (a3/a1) times the first equation from the third equation.

Then we get the new equations as:

a1x + b1y + c1z = d1
b'2y + c'2z = d'2
c''3z = d''3

The elimination procedure is continued until only one unknown remains in the last equation. After its value is determined, the procedure is stopped. Now, Gauss Elimination in C uses back substitution to get the values of x, y and z as:

z= d''3 / c''3
y=(d'2 – c'2z) / b'2
x=(d1- c1z- b1y)/ a1.

Algorithm:

1. Start
2. Declare the variables and read the order of the matrix n.
3. Take the coefficients of the linear equation as:
   Do for k=1 to n
   Do for j=1 to n+1
   Read a[k][j]
   End for j
   End for k
4. Do for k=1 to n-1
   Do for i=k+1 to n

a[i][j] = a[i][j] – a[i][k] /a[k][k] * a[k][j]

End for j

End for i

End for k

5. Compute x[n] = a[n][n+1]/a[n][n]
6. Do for k=n-1 to 1

   sum = 0

   Do for j=k+1 to n

   sum = sum + a[k][j] * x[j]

   End for j

   x[k] = 1/a[k][k] * (a[k][n+1] – sum)

   End for k
7. Display the result x[k]
8. Stop

/* Program to to implement Gauss elimination method */

```
#include<stdio.h>
int main()
{
   inti,j,k,n;
   float A[20][20],c,x[10],sum=0.0;
   printf("\nEnter the order of matrix: ");
   scanf("%d",&n);
   printf("\nEnter the elements of augmented matrix row-wise:\n\n");
   for(i=1; i<=n; i++)
   {
      for(j=1; j<=(n+1); j++)
      {
         printf("A[%d][%d] : ", i,j);
         scanf("%f",&A[i][j]);
      }
   }
   for(j=1; j<=n; j++) /* loop for the generation of upper triangular matrix*/
   {
      for(i=1; i<=n; i++)
      {
         if(i>j)
         {
            c=A[i][j]/A[j][j];
            for(k=1; k<=n+1; k++)
            {
               A[i][k]=A[i][k]-c*A[j][k];
            }
         }
      }
   }
   x[n]=A[n][n+1]/A[n][n];
   /* this loop is for backward substitution*/
   for(i=n-1; i>=1; i--)
   {
```

```
    for(j=i+1; j<=n; j++)
    {
       sum=sum+A[i][j]*x[j];
    }
    x[i]=(A[i][n+1]-sum)/A[i][i];
  }
  printf("\nThe solution is: \n");
  for(i=1; i<=n; i++)
  {
    printf("\nx%d=%f\t",i,x[i]); /* x1, x2, x3 are the required solutions*/
  }
  return(0);
```

OUTPUT :

```
No of Equtions : 3
Enter Coefficients of Eqution
4 3 -2
1 1 1
3 -2 1
Enter Constant value
5 3 2
Eliminated matrix as :-
4.00 3.00 -2.00 5.00
0.00 0.25 1.50 1.75
0.00 0.00 28.00 28.00
Solution :
X3 = 1.00
X2 = 1.00
X1 = 1.00
```

Experiment No:3(b) Gauss-Seidel iterations.
AIM: Write a C Program to implement Gauss-Seidel iterations method.

Description:
The program for Gauss-Seidel method in C works by following the steps listed below:

When the program is executed, first of all it asks for the value of elements of the augmented matrix row wise.
Then, the program asks for allowed error and maximum number of iteration to which the calculations are to be done. The number of iterations required depends upon the degree of accuracy.
The program assumes initial or approximate solution as y=0 and z=0 and new value of x which is used to calculate new values of y and z using the following expressions:
x= 1/a1 (d1-b1y-c1z)
y=1/b2 (d2-a2x-c2z)
z=1/c3 (d3-a3x-b3y)

Algorithm:

1. Start
2. Declare the variables and read the order of the matrix n
3. Read the stopping criteria er
4. Read the coefficients aim as

   Do for i=1 to n

   Do for j=1 to n

Repeat for j

Repeat for i

5. Read the coefficients b[i] for i=1 to n
6. Initialize x0[i] = 0 for i=1 to n
7. Set key=0
8. For i=1 to n

   Set sum = b[i]

   For j=1 to n

   If (j not equal to i)

   Set sum = sum – a[i][j] * x0[j]

   Repeat j

   x[i] = sum/a[i][i]

   If absolute value of ((x[i] – x0[i]) / x[i]) >er, then

   Set key = 1

   Set x0[i] = x[i]

   Repeat i
9. If key = 1, then

   Goto step 6

   Otherwise print results

.

/* Program to implement Gauss-Seidel iterations method. */

```
#include<stdio.h>
#include<math.h>
#define X 2
main()
{
   float x[X][X+1],a[X], ae, max,t,s,e;
   inti,j,r,mxit;
   for(i=0;i<X;i++) a[i]=0;
   puts(" Eneter the elemrnts of augmented matrix rowwise\n");
   for(i=0;i<X;i++)
   {
   for(j=0;j<X+1;j++)
   {
   scanf("%f",&x[i][j]);
   }
   }
   printf(" Eneter the allowed error and maximum number of iteration: ");
   scanf("%f%d",&ae,&mxit);
   printf("Iteration\tx[1]\tx[2]\n");
   for(r=1;r<=mxit;r++)
   {
      max=0;
      for(i=0;i<X;i++)
      {
         s=0;
```

```
            if(j!=i) s+=x[i][j]*a[j];
            t=(x[i][X]-s)/x[i][i];
            e=fabs(a[i]-t);
            a[i]=t;
        }
        printf(" %5d\t",r);
        for(i=0;i<X;i++)
        printf(" %9.4f\t",a[i]);
        printf("\n");
        if(max<ae)
        {
            printf(" Converses in %3d iteration\n", r);
            for(i=0;i<X;i++)
            printf("a[%3d]=%7.4f\n", i+1,a[i]);
            return 0;
        }

        }
    }
```

OUTPUT :

```
Enter the number of equations: 3
Enter the co-efficients of the equations:
a[1][1]= 2
a[1][2]= 1
a[1][3]= 1
a[1][4]= 5
a[2][1]= 3
a[2][2]= 5
a[2][3]= 2
a[2][4]= 15
a[3][1]= 2
a[3][2]= 1
a[3][3]= 4
a[3][4]= 8

x[1] =2.500000
x[2] =1.500000
x[3] =0.375000

x[1] =1.562500
x[2] =1.912500
x[3] =0.740625

x[1] =1.173437
x[2] =1.999688
x[3] =0.913359

x[1] =1.043477
x[2] =2.008570
x[3] =0.976119

x[1] =1.007655
x[2] =2.004959
x[3] =0.994933

x[1] =1.000054
```

x[3] =0.999474

converges to solution
x[1]=1.000054
x[2]=2.001995
x[3]=0.999474

Experiment No: 4(a)Regular-falsi

AIM: Write a program to implement Regular-falsi method.

Description:
The C Program for regulafalsi method requires two initial guesses of opposite nature. Like the secant method, interpolation is done to find the new values for successive iterations, but in this method one interval always remains constant.

The programming effort for RegulaFalsi or False Position Method in C language is simple and easy. The convergence is of first order and it is guaranteed. In manual approach, the method of false position may be slow, but it is found superior to the bisection method.

- tr – a counter which keeps track of the no. of iterations performed
- maxmitr – maximum number of iterations to be performed
- x0, x1 – the limits within which the root lies
- x2 – the value of root at nth iteration
- x3 – the value of root at (n+1)th iteration
- allerr – allowed error
- x – value of root at nth iteration in the regula function
- f(x0), f(x1) – the values of f(x) at x0 and x1 respectively

f(x) = cos(x) – x*e^x

Algorithm :

1. Start
2. Read values of x0, x1 and e
   *Here x0 and x1 are the two initial guesses
   e is the degree of accuracy or the absolute error i.e. the stopping criteria*
3. Computer function values f(x0) and f(x1)
4. Check whether the product of f(x0) and f(x1) is negative or not.
   If it is positive take another initial guesses.
   If it is negative then goto step 5.
5. Determine:
   x = [x0*f(x1) – x1*f(x0)] / (f(x1) – f(x0))
6. Check whether the product of f(x1) and f(x) is negative or not.
   If it is negative, then assign x0 = x;
   If it is positive, assign x1 = x;
7. Check whether the value of f(x) is greater than 0.00001 or not.
   If yes, goto step 5.
   If no, goto step 8.
   *Here the value 0.00001 is the desired degree of accuracy, and hence the stopping

8. Display the root as x.
9. Stop

/* Program to implement Regular-falsi method */

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
   returncos(x) - x*exp(x);
}
voidregula (float *x, float x0, float x1, float fx0, float fx1, int *itr)
{
   *x = x0 - ((x1 - x0) / (fx1 - fx0))*fx0;
   ++(*itr);
   printf("Iteration no. %3d X = %7.5f \n", *itr, *x);
}
void main ()
{
   intitr = 0, maxmitr;
   float x0,x1,x2,x3,allerr;
   printf("\nEnter the values of x0, x1, allowed error and maximum iterations:\n");
   scanf("%f %f %f %d", &x0, &x1, &allerr, &maxmitr);
   regula (&x2, x0, x1, f(x0), f(x1), &itr);
   do
   {
      if (f(x0)*f(x2) < 0)
         x1=x2;
      else
         x0=x2;
      regula (&x3, x0, x1, f(x0), f(x1), &itr);
      if (fabs(x3-x2) <allerr)
      {
         printf("After %d iterations, root = %6.4f\n", itr, x3);
         return 0;
      }
      x2=x3;
   }
   while (itr<maxmitr);
   printf("Solution does not converge or iterations not sufficient:\n");
   return 1;
}
```

OUTPUT :

```
Enter the value of x0: -1

Enter the value of x1: 1

_____________________________________________________________

x0        x1        x2        f0      f1      f2
_____________________________________________________________

-1.000000 1.000000 0.513434 -4.540302 1.459698 -0.330761
0.513434  1.000000 0.603320 -0.330761 1.459698 -0.013497
0.603320  1.000000 0.606954 -0.013497 1.459698 -0.000527
0.606954  1.000000 0.607096 -0.000527 1.459698 -0.000021
_____________________________________________________________
```

Experiment No: 4(b) Newton Raphson methods
AIM: Write a program to implement Newton Raphson methods.

Algorithm:

1. Start
2. Read x, e, n, d
   *x is the initial guess
   e is the absolute error i.e the desired degree of accuracy
   n is for operating loop
   d is for checking slope*
3. Do for i =1 to n in step of 2
4. f = f(x)
5. f1 = f'(x)
6. If ([f1] < d), then display too small slope and goto 11.
   [ ] is used as modulus sign
7. x1 = x – f/f1
8. If ([(x1 – x)/x1] < e), the display the root as x1 and goto 11.
   [ ] is used as modulus sign
9. x = x1 and end loop
10. Display method does not converge due to oscillation.
11. Stop

/* Program to implement Newton Raphson methods */

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
   return x*log10(x) - 1.2;
}
floatdf (float x)
{
   return log10(x) + 0.43429;
}
void main()
{
   intitr, maxmitr;
   float h, x0, x1, allerr;
   printf("\nEnter x0, allowed error and maximum iterations\n");
   scanf("%f %f %d", &x0, &allerr, &maxmitr);
   for (itr=1; itr<=maxmitr; itr++)
```

```
        x1=x0-h;
        printf(" At Iteration no. %3d, x = %9.6f\n", itr, x1);
        if (fabs(h) <allerr)
        {
           printf("After %3d iterations, root = %8.6f\n", itr, x1);
           return 0;
        }
        x0=x1;
    }
    printf(" The required solution does not converge or iterations are insufficient\n");
    return 1;
}
```

OUTPUT :

```
ENTER THE TOTAL NO. OF POWER:::: 3

x^0::-3

x^1::-1

x^2::0

x^3::1

    THE POLYNOMIAL IS :::  1x^3 0x^2 -1x^1 -3x^0
    INTIAL X1---->3
 ITERATION    X1    FX1    F'X1
  1        2.192  21.000 26.000
 2        1.794  5.344  13.419
 3        1.681  0.980  8.656
 4        1.672  0.068  7.475
 5        1.672  0.000  7.384
THE ROOT OF EQUATION IS 1.671700
```

Experiment No:5(a)Euler' s methods

AIM: Write a program to simulate Euler' s method.

Description:

Solving an ordinary differential equation or initial value problem means finding a clear expression for y in terms of a finite number of elementary functions of x. Euler's method is one of the simplest method for the numerical solution of such equation or problem. This **C program for Euler's method** considers an ordinary differential equations, and the initial values of x and y are known. Mathematically, here, the curve of solution is approximated by a sequence of short lines i.e. by the tangent line in each interval. Using these information, the value of the value of 'y_n' corresponding to the value of 'x_n' is to determined by dividing the length $(x_n - x)$ into n strips.

Therefore, strip width= $(x_n - x)/n$ and $x_n=x_0+ nh$.

Again, if m be the slope of the curve at point, $y_{1=} y_{0+} m(x_0 , y_o)h$.

Similarly, values of all the intermediate y can be found out.

Below is a source code for **Euler's method in C** to solve the ordinary differential equation **dy/dx = x+y**. It asks for the value of of x_0 , y_0 , x_n and h. The value of slope at different points is calculated using the function 'fun'.

The values of y are calculated in while loop which runs till the initial value of x is not equal to the final value. All the values of 'y' at corresponding 'x' are shown in the output screen.

dy/dx = x+y

Algorithm:

1. Start

2. Define function
3. Get the values of x0, y0, h and xn
   *Here x0 and y0 are the initial conditions
   h is the interval
   xn is the required value
4. n = (xn – x0)/h + 1
5. Start loop from i=1 to n
6. y = y0 + h*f(x0,y0)
   x = x + h
7. Print values of y0 and x0
8. Check if x <xn
   If yes, assign x0 = x and y0 = y
   If no, goto 9.
9. End loop i
10. Stop

/* Program to simulate Euler' s method */

```
#include<stdio.h>
float fun(float x,float y)
{
   float f;
   f=x+y;
   return f;
}
main()
{
   floata,b,x,y,h,t,k;
   printf("\nEnter x0,y0,h,xn: ");
   scanf("%f%f%f%f",&a,&b,&h,&t);
   x=a;
   y=b;
   printf("\n  x\t  y\n");
   while(x<=t)
   {
      k=h*fun(x,y);
      y=y+k;
      x=x+h;
      printf("%0.3f\t%0.3f\n",x,y);
   }
}
```

OUTPUT :

```
Enter the value of range: 1 1.5
Enter the value of y1: 5
Enter the h: 0.1
  y1 = 5.000

  x = 1.000 => y2 = 5.500

  x = 1.100 => y3 = 6.105
```

x = 1.300 => y5 = 7.726

x = 1.400 => y6 = 8.808

x = 1.500 => y7 = 10.129

Experiment No:5(b)Runga-Kutta methods
AIM: Write a program to simulate Runga-Kutta methods.

/* Program to simulate Runga-Kutta methods*/

```
#include<stdio.h>
#include<math.h>
float f(float x,float y);
int main()
{
   float x0,y0,m1,m2,m3,m4,m,y,x,h,xn;
   printf("Enter x0,y0,xn,h:");
   scanf("%f %f %f %f",&x0,&y0,&xn,&h);
   x=x0;
   y=y0;
   printf("\n\nX\t\tY\n");
   while(x<xn)
   {
      m1=f(x0,y0);
      m2=f((x0+h/2.0),(y0+m1*h/2.0));
      m3=f((x0+h/2.0),(y0+m2*h/2.0));
      m4=f((x0+h),(y0+m3*h));
      m=((m1+2*m2+2*m3+m4)/6);
      y=y+m*h;
      x=x+h;
      printf("%f\t%f\n",x,y);
   }
}
float f(float x,float y)
{
   float m;
   m=(x-y)/(x+y);
   return m;
}
```

OUTPUT:

Enter the value of x0: 0

Enter the value of y0: 2

Enter the value of h: 0.05

Enter the value of last point: 0.1

k1 = 0.1000

k2 = 0.1025

y(0.0500) = 2.101

k1 = 0.1026

y(0.1000) = 2.205

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Computer NetworksLab
Course Code: MCA392
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:
This practical course provides students with hands on training regarding the design, troubleshooting, modeling and evaluation of computer networks. In this course, students are going to experiment in a real and simulation based test-bed networking environment, and learn about network design and troubleshooting topics and tools such as: network addressing, Address Resolution Protocol, basic troubleshooting tools (like ping, ICMP), IP routing (e.g. RIP), TCP and UDP,DHCP,ACL and many others. Student will have the opportunity to build some simple networking models using the tool and perform simulations that will help them evaluate their design approaches and expected network performance.
Learning Outcomes: The students will have a detailed knowledge network topology, Local area network, IP addressing, familiarization with network simulator, idea about networking devices, network cable and connectors, different types routing protocols, concept of remote access and different types of application layer protocol. Upon the completion of Computer network practical course, the student will be able to:

- **Learn** various network commands.
- **Understand** and implement basic of Network and Network Topology.
- **To get** idea about IP addressing schemes.
- **Understand** the benefits of network.
- **Configure** and simulate various protocols.
- **Access** remote desktop.
- **Connect** to different computer using LAN.
- **Understand** the concepts of access control.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1: Study of different types of Network cables and practically implements the cross-wired cable and straight through cable using clamping tool.
Exercise No. 2: Familiarization with some network devices.
Exercise No. 3: Study of Network IP.
Exercise No. 4: Connect the computers in LAN.
Exercise No. 5: Introduction to Packet Tracer.
Exercise No. 6: Configure network topology using packet tracer.
Exercise No. 7: Configure network topology using packet tracer to find the routing path by IPRoute Command.
Exercise No. 8: Network Configuration using distance vector routing protocol.
Exercise No. 9: Configuration of DHCP Protocol
Exercise No. 10: Telnet Configuration.
Exercise No. 11: Configuration of Access Control List.

Text Book:
1. B. A. Forouzan – "Data Communications and Networking (3rd Ed.) " – TMH
Reference Book:
1. Authorized Self-Study Guide "Interconnecting Cisco Network Devices, Part 1(ICND1), 2nd Edition, January, 2008.

Recommended Systems/Software Requirements:
1. CAT-5/CAT-6 Cables, RJ 45, Cutter, Clamping Tool, Router , Switch and Hub.
2. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

Experiment No: 1 STUDY OF DIFFERENT TYPES OF NETWORK CABLES
Aim: Study of different types of Network cables and practically implements the cross-wired cable and straight through cable using clamping tool.
Apparatus: RJ-45 connector, Clamping Tool, Twisted pair Cable
Description:
RJ-45: Registered Jack 45 (RJ45) is a standard type of physical connector for network cables. RJ45 connectors are most commonly seen with Ethernet cables and networks.

Two standard RJ45 pinouts define the arrangement of the individual eight wires needed when attaching connectors to a cable - the T568A and T568B standards. Both follow a convention of coating individual wires in one of five colors - brown, green, orange, blue and white - with certain stripe and solid combinations.

Following these conventions is essential when building cables to ensure electrical compatibility with other equipment. For historical reasons, T568B has become the more popular standard. The table below summarizes this color coding.

Table-1

T568B / T568A Pinouts		
Pin	**T568B**	**T568A**
1	white with orange stripe	white with green stripe
2	orange	Green
3	with with green stripe	white with orange stripe
4	blue	Blue
5	white with blue stripe	white with blue stripe
6	green	Orange
7	white with brown stripe	white with brown stripe
8	Brown	Brown

Figure-1: RJ-45

Clamping Tool: A clamping tool is a fastening device used to hold or secure objects tightly together to prevent movement or separation through the application of inward pressure. It is used to fasten RJ-45 with cable tightly.

Figure-2: Clamping Tool

Twisted Pair Cable: Twisted pair is the ordinary copper wire that connects home and many business computers to the telephone company. To reduce crosstalk or electromagnetic induction between pairs of wires, two insulated copper wires are twisted around each other. Each connection on twisted pair requires both wires. Since some telephone sets or desktop locations require multiple connections, twisted pair is sometimes installed in two or more pairs, all within a single cable. For some business locations, twisted pair is enclosed in a shield that functions as a ground. This is known as shielded twisted pair (STP). Ordinary wire to the home is unshielded twisted pair (UTP).

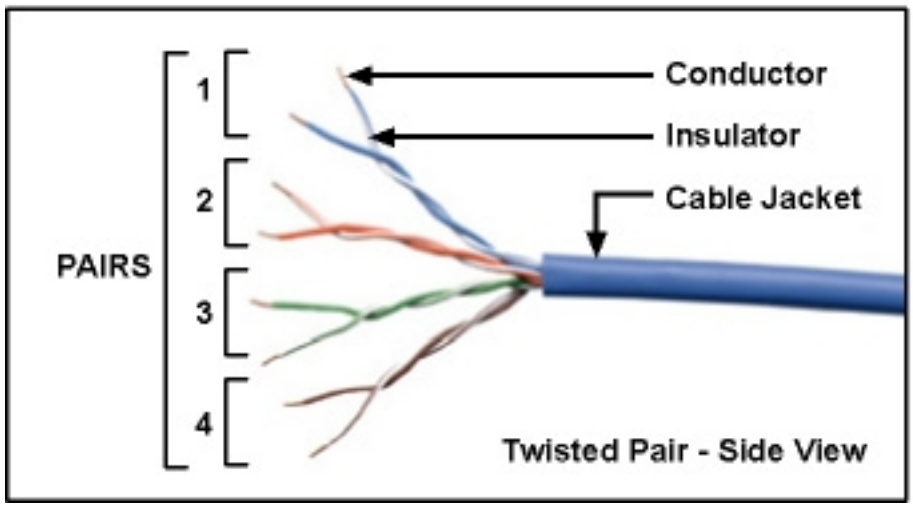

Figure-3: Twisted Pair Cable

Procedure: To do these practical following steps should be done:
1. Start by stripping about of the plastic jacket off the end of the cable. Be very careful at this point, as to not nick or cut into the wires, which are inside. Doing so could alter the characteristics of your cable, or even worse render is useless. Check the wires, one more time for nicks or cuts. If there are any, just whack the whole end off, and start over.
2. Spread the wires apart, but be sure to hold onto the base of the jacket with your other hand. You do not want the wires to become untwisted down inside the jacket. Category 5/6 cable must only have ½ of an inch of 'untwisted' wire at the end; otherwise it will be 'out of spec'. At this point, you obviously have A LOT more than ½ of an inch of un-twisted wire.
3. You have 2 end jacks, which must be installed on your cable. If you are using a pre-made cable, with one of the ends whacked off, you only have one end to install-the crossed over end. Below are two diagrams, which show how you need to arrange the cables for each type of cable end. Decide at this point which end you are making and examine the associated picture below. Figure-4 shows you how to prepare cross wired connection. Figure-5 shows you how to prepare straight through wired connection.

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

Figure-4: Color Combination (T568B) Table

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

Figure-5: Color Combination Table

Experiment No: 2 FAMILIARIZATION WITH SOME NETWORK DEVICES
Aim: Study of following network devices

- Switch
- Router
- Hub
- Repeater

Apparatus: Hardware needed
Procedure: Following should be done to understand this practical
1. Switch: A network switch or switching hub is a computer networking device that connects network segments. The term commonly refers to a network bridge that processes and routes data

at the data link layer of the OSI model. Switches that additionally process data at the network layer are often referred to as Layer 3 switches or multilayer switches.

2. Router: A router is an electronic device that interconnects two or more computer networks, and selectively interchanges packets of data between them. Each data packet contains address information that a router can use to determine if the source and destination are on the same network, or if the data packet must be transferred from one network to another. Where multiple routers are used in a range collection of interconnected networks, the routers exchange information about target system addresses, so that each router can build up a table showing the preferred paths between any two systems on the interconnected networks.

3. Repeater: Functioning at Physical Layer a repeater is an electronic device that receives a signal and retransmits it at a higher level and /or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances. Repeater have two ports, so cannot be use to connect for more than two devices.

4. Hub: An Ethernet hub, active hub, network hub, repeater hub, hub or concentrator is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and making them act as single network segment. Hubs work at physical layer of the OSI model. The device is a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision.

Experiment No: 3 STUDY OF NETWORK IP
Aim: Study of IP address of network.

- Classification of IP address
- Sub netting
- Super netting

Description: In the most widely installed level of the Internet Protocol (IP) today, an IP address is a 32-bit number that identifies each sender or receiver of information that is sent in packets across the Internet. When you request an HTML page or send e-mail, the Internet Protocol part of TCP/IP includes your IP address in the message (actually, in each of the packets if more than one is required) and sends it to the IP address that is obtained by looking up the domain name in the Uniform Resource Locator you requested or in the e-mail address you're sending a note to. At the other end, the recipient can see the IP address of the Web page requestor or the e-mail sender and can respond by sending another message using the IP address it received.

An IP address has two parts: the identifier of a particular network on the Internet and an identifier of the particular device (which can be a server or a workstation) within that network. On the Internet itself - that is, between the router that move packets from one point to another along the route - only the network part of the address is looked at.

The Network Part of the IP Address:

The Internet is really the interconnection of many individual networks (it's sometimes referred to as an internetwork). So the Internet Protocol (IP) is basically the set of rules for one network communicating with any other (or occasionally, for broadcast messages, all other networks). Each network must know its own address on the Internet and that of any other networks with which it communicates. To be part of the Internet, an organization needs an Internet network number, which it can request from the Network Information Center (NIC). This unique network number is included in any packet sent out of the network onto the Internet.

The Local or Host Part of the IP Address:
In addition to the network address or number, information is needed about which specific machine or host in a network is sending or receiving a message. So the IP address needs both the unique network number and a host number (which is unique within the network). (The host number is sometimes called a local or machine address.)

Part of the local address can identify a subnetwork or subnet address, which makes it easier for a network that is divided into several physical subnetworks (for examples, several different local area networks or) to handle many devices.

Apparatus: N/A
Procedure: Following is required to be study under this practical.
Classification of IP address: As show in figure-6 we teach how the IP addresses are classified and when they are used.

Class	Address Range	Supports
Class A	1.0.0.1 to 126.255.255.254	Supports 16 million hosts on each of 127 networks.
Class B	128.1.0.1 to 191.255.255.254	Supports 65,000 hosts on each of 16,000 networks.
Class C	192.0.1.1 to 223.255.254.254	Supports 254 hosts on each of 2 million networks.
Class D	224.0.0.0 to 239.255.255.255	Reserved for multicast groups.
Class E	240.0.0.0 to 254.255.255.254	Reserved.

Figure-6: Classification of IP address

Sub netting: Why we develop sub netting and how to calculate subnet mask to identify subnet address.
Super netting: Why we develop super netting and how to calculate super net mask and how to identify super net address.
Command: Open your computer →go to start button →write run then click on run →write 'CMD' then a black screen will appear then type ipconfig on the screen then you can see the IP address as well as subnet mask.

Input:
Command: C:>ipconfig

Output:

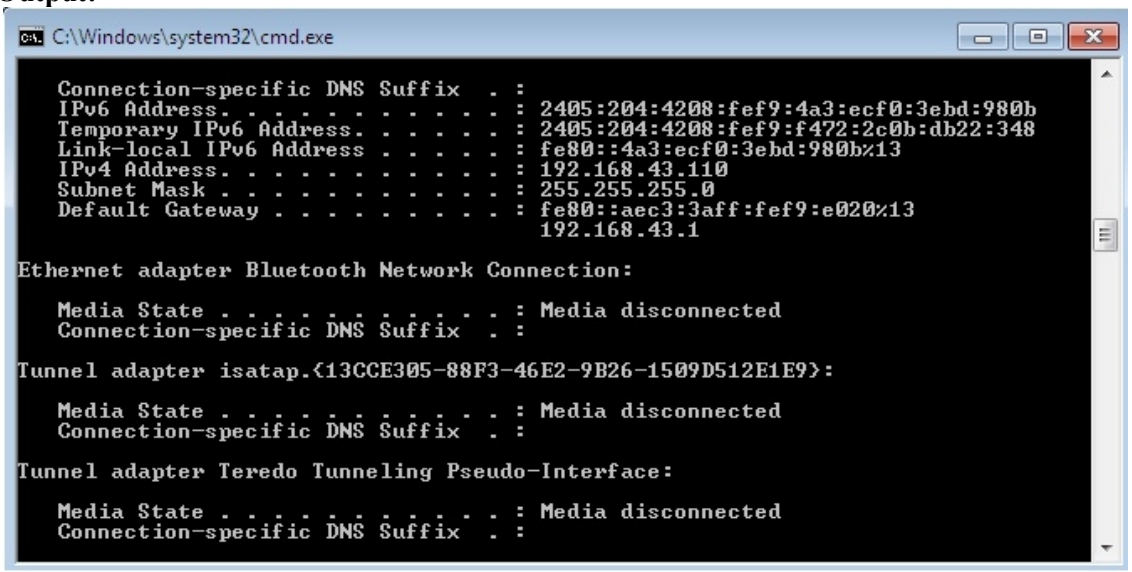

Figure-7: Output Screen

Experiment No-4 CONNECT THE COMPUTERS IN LOCAL AREA NETWORK

Description: A local-area network (LAN) is a computer network that spans a relatively small area. Most often, a LAN is confined to a single room, building or group of buildings, however, one LAN can be connected to other LANs over any distance via telephone lines and radio waves. A system of LANs connected in this way is called a wide-area network (WAN).

Nodes on a Local Area Network:

Most LANs connect workstations and personal computers. Each node (individual computer) in a LAN has its own CPU with which it executes programs, but it also is able to access data and devices anywhere on the LAN. This means that many users can share expensive devices, such as laser printers, as well as data. Users can also use the LAN to communicate with each other, by sending email or engaging in chat sessions.

LANs are capable of transmitting data at very fast rates, much faster than data can be transmitted over a telephone line; but the distances are limited and there is also a limit on the number of computers that can be attached to a single LAN.

Types of Local-Area Networks:

There are many different types of LANs, with **Ethernets** being the most common for PCs. Most Apple Macintosh networks are based on **Apple's AppleTalk** network system, which is built into Macintosh computers.

Aim: Connect two different computers in Local Area Network

Procedure:

On the host computer: On the host computer, follow these steps to share the internet connection:

1. Log on to the host computer as **Administrator** or as **Owner**.
2. Click **Start,** and then click **Control Panel**.
3. Click **Network and Internet Connections**.
4. Click **Network Connections**.
5. Right-click the connection that you use to connect to the internet. For example, if you connect to the Internet by using a modem, right-click the connection that you want under Dial-up/other network available.
6. Click **Properties**.
7. Click the **Advanced** Tab.
8. Under **Internet Connection Sharing**, select the **Allow other network users to connect through this computer's Internet connection** check box.
9. If you are sharing a dial-up Internet connection, select the **Establish a dial-up connection whenever a computer on my network attempts to access the Internet** check box if you want to permit your computer to automatically connect to the Internet.
10. Click **OK.** You receive the following message:

When Internet Connection Sharing is enabled, your LAN adapter will be set to use IP address 192.168.10.1. Your computer may lose connectivity with other computers on your network. If these other computers have static IP address, it is a good idea to set them to obtain their IP addresses automatically. Are you sure you want to enable Internet Connection sharing?

11. Click **YES.**

The connection to the Internet is shared to other computers on the local area network (LAN)

The network adapter that is connected to the LAN is configured with a static IP address of 192.168.0.1 and a subnet mask of 255.255.255.0.

On the client computer: To connect to the Internet by using the shared connection, you must confirm the LAN adapter IP configuration, and then configure the client computer. To confirm the LAN adapter IP configuration, follow these steps:

1. Log on to the client computer as **Administrator** or **Owner**.
2. Click **Start**, and then click **Control Panel**.
3. Click **Network and Internet Connections**.

4. Click **Network Connections.**
5. Right-click **Local Area Connection** and then click **properties.**
6. Click on **General** tab, click **Internet Protocol (TCP/IP)** in the connection uses the following items list, and then click **Properties**.
7. In the **Internet Protocol (TCP/IP) Properties** dialog box, click **Obtain an IP address automatically** (if it is not already selected), and then click **OK.**
Note: You can also assign unique static IP address in the range of 192.168.0.2 to 192.168.0.254. For example, you can assign the following static IP address, subnet mask, and default gateway:
IP Address 192.168.31.202
Subnet mask: 255.255.255.0
Default gateway 192.168.31.1
8. In the **Local Area Connection Properties** dialog box, click **OK.**
9. Quit **Control Panel**.

Experiment No-5 INTRODUCTION TO PACKET TRACER
Aim: Study of basic network command and Network configuration commands.
Description: Packet Tracer is a cross-platform visual simulation program designed by Cisco Systems that allows users to create network topologies and imitate modern computer networks. The software allows users to simulate the configuration of Cisco routers and switches using a simulated command line interface. Packet Tracer makes use of a drag and drop user interface, allowing users to add and remove simulated network devices as they see fit. The software is mainly focused towards Certified Cisco Network Associate Academy students as an educational tool for helping them learn fundamental CCNA concepts. Students enrolled in a CCNA Academy program can freely download and use the tool free of charge for educational use.

In addition to simulating certain aspects of computer networks, Packet Tracer can also be used for collaboration. As of Packet Tracer 5.0, Packet Tracer supports a multi-user system that enables multiple users to connect multiple topologies together over a computer network. Packet Tracer also allows instructors to create activities that students have to complete. Packet Tracer is often used in educational settings as a learning aid. Cisco Systems claims that Packet Tracer is useful for network experimentation

Apparatus (Software): Command Prompt and Packet Tracer.
Procedure: To do this experiment – follow these steps:
In this experiment students have to understand basic networking commands e.g ping, tracert etc. All commands related to network configuration which includes how to switch to privilege mode and normal mode and how to configure router interface and how to save this configuration to flash memory or permanent memory.

This commands includes

- Configuring the Router commands
- General Commands to configure network
- Privileged Mode commands of a router
- Router Processes & Statistics
- IP Commands
- Other IP Commands e.g. show ip route etc.

Ping: ping sends an ICMP ECHO_REQUEST packet to the specified host. If the host responds, you get an ICMP packet back. Sound strange? Well, you can "ping" an IP address to see if a machine is alive. If there is no response, you know something is wrong.

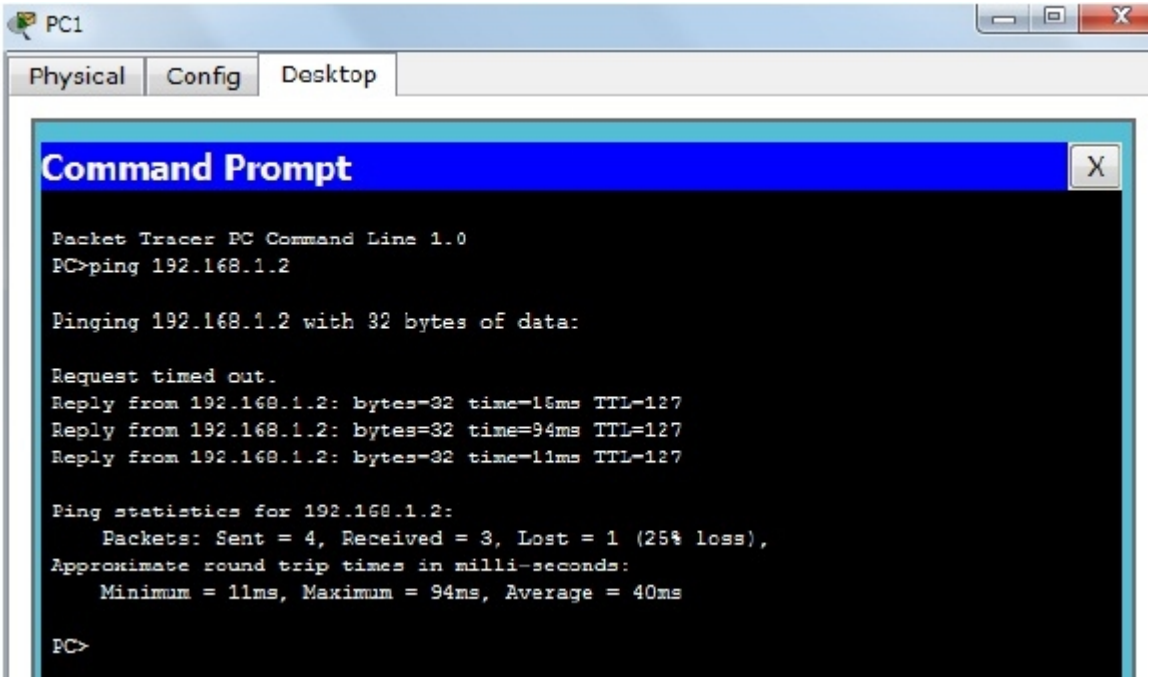

Figure-8: Command Prompt of PC1

Traceroute: Tracert is a command which can show you the path a packet of information takes from your computer to one you specify. It will list all the routers it passes through until it reaches its destination, or fails to and is discarded. In addition to this, it will tell you how long each 'hop' from router to router takes.

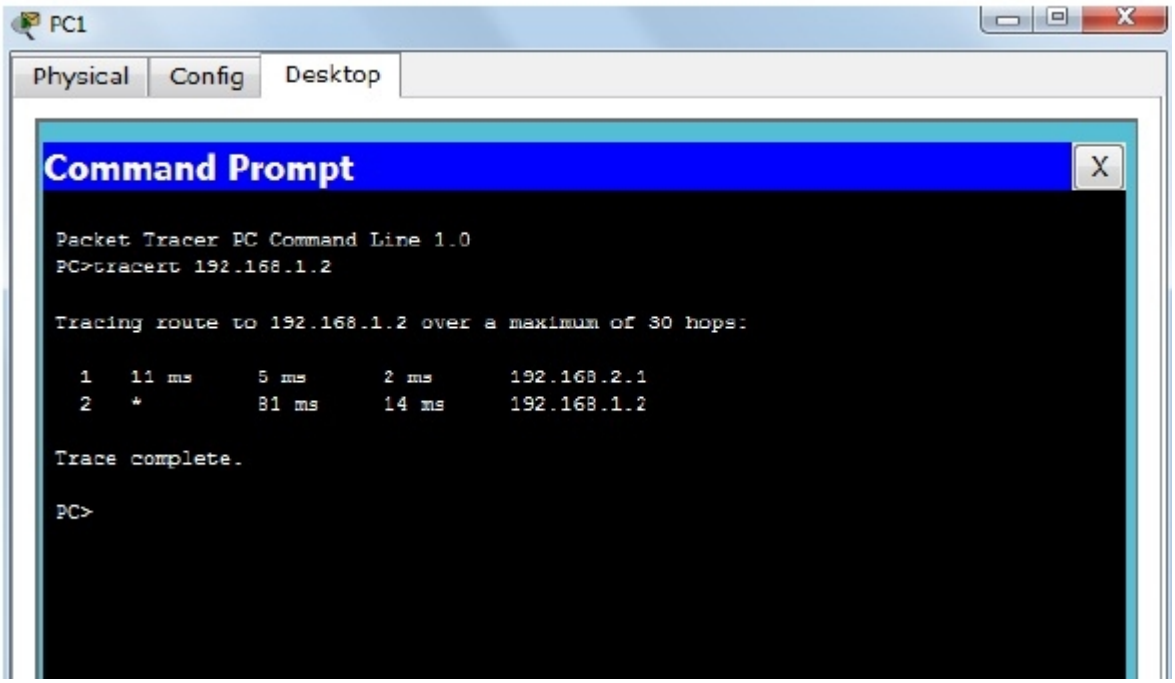

Figure-9: Command Prompt of PC1

Nslookup: Displays information from Domain Name System (DNS) name servers.
NOTE: If you write the command as above it shows as default your pc's server name firstly.
Pathping: A better version of tracert that gives you statistics about packet lost and latency.

Getting Help:
In any command mode, you can get a list of available commands by entering a question mark '?'.
Router>?
To obtain a list of commands that begin with a particular character sequence, type in those characters followed immediately by the question mark (?).
Router# co?
Configure connect copy
To list keywords or arguments, enter a question mark in place of a keyword or argument. Include a space before the question mark.
Router# configure?
Memory Configure from NV memory
Network configure from a TFTP network host
Terminal configure from the terminal
You can also abbreviate commands and keywords by entering characters to make the command unique from other commands. For example, you can abbreviate the **show** command to **sh.**

Configuration Files:
Any time you make changes to the router configuration, you must save the changes to memory because if you do not they will be lost if there is a system reload or power outage. There are two types of configuration files: the running (currently operating) configuration and the startup configuration.
Use the following privileged mode commands to work with configuration files.
Configuration terminal-modify the running configuration manually from the terminal.

- Show running-config- display the running configuration.
- Show startup-config- display the startup configuration
- Copy running-config startup-config- copy the running configuration to the startup configuration.
- Copy startup-config running- config- copy the start up configuration to the running configuration.
- Erase startup-config- erase the startup-configuration in NVRAM.

IP Address Configuration:
Take the following steps to configure the IP address of an interface.
Step-1: Enter privileged EXEC mode:
Router> **enable** password
Step-2: Enter the **configure terminal** command to enter global configuration mode.
Router# **config terminal**
Step-3: Enter the **interface** type slot/port or **interface** type port to enter the interface configuration mode.
Example:
Router(config)#**interface Ethernet 0/1**
Step-4: Enter the IP address and subnet mask of the interface using the **ip address** ipaddress subnetmask command.

Example:
Router(config-if)#**ip address 192.168.10.1 255.255.255.0**
Step-5: Exit the configuration mode by pressing Ctrl-Z
Router(config-if)#Ctrl-Z

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Experiment No-6 CONFIGURE NETWORK TOPOLOGY USING PACKET TRACER
Aim: To configure network topology using packet tracer software
Description: Network Topology refers to layout of a network and how different nodes in a network are connected to each other and how they communicate. Topologies are either physical (the physical layout of devices on a network) or logical (the way that the signals act on the network media, or the way that the data passes through the network from one device to the next).
Some of the topologies are:

- Bus
- Mesh
- Star
- Ring
- Hybrid

Apparatus (Software): Packet Tracer 5.5.
Procedure: To implement this practical following network topology is required to be configured using the commands discussed above.
After configuring the given network a packet should be ping from any one machine to another.

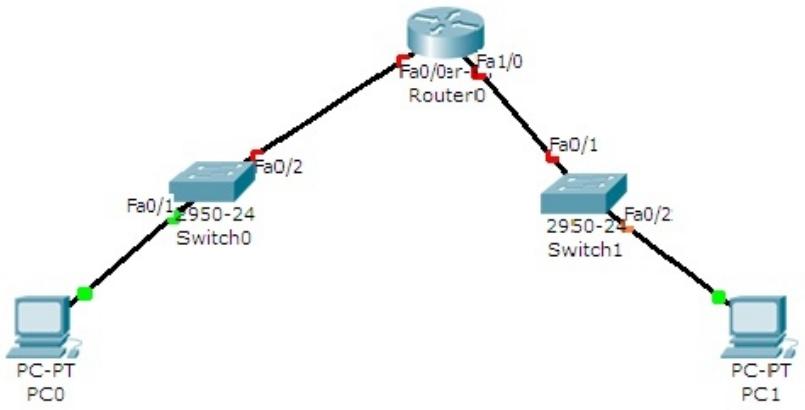

Figure-10: Network Topology

Router-0 Configuration Command:
Continue with configuration dialog? [yes/ no]: no

Press RETURN to get started!
Router>
Router>Enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# hostname router0
router0(config) # interface fastethernet 0/0 **# Configuration of 0/0 port#**
router0(config-if) # ip address 192.168.1.1 255.255.255.0
router0(config-if) # no shutdown
router0(config-if)# exit
router0(config)# interface fastethernet 1/0 **# Configuration of 1/0 port#**
router0(config-if)# ip address 192.168.2.1 255.255.255.0
router0(config-if)# no shutdown
router0(config-if)# exit

router0(config)# exit
router0# show running-config
Building configuration….
Current configuration: 356 bytes
:
:
:
Version 12.4
Etc etc…….
Ip address 192.168.1.1 255.255.255.0
:
: interface VLAN
No ip address
:
:
:
End
router0#
router0#
router0# copy running-config startup-config
Destination filename[startup-config]?
Building configuration…
[OK]
router0#

IP Configuration of PC-0:

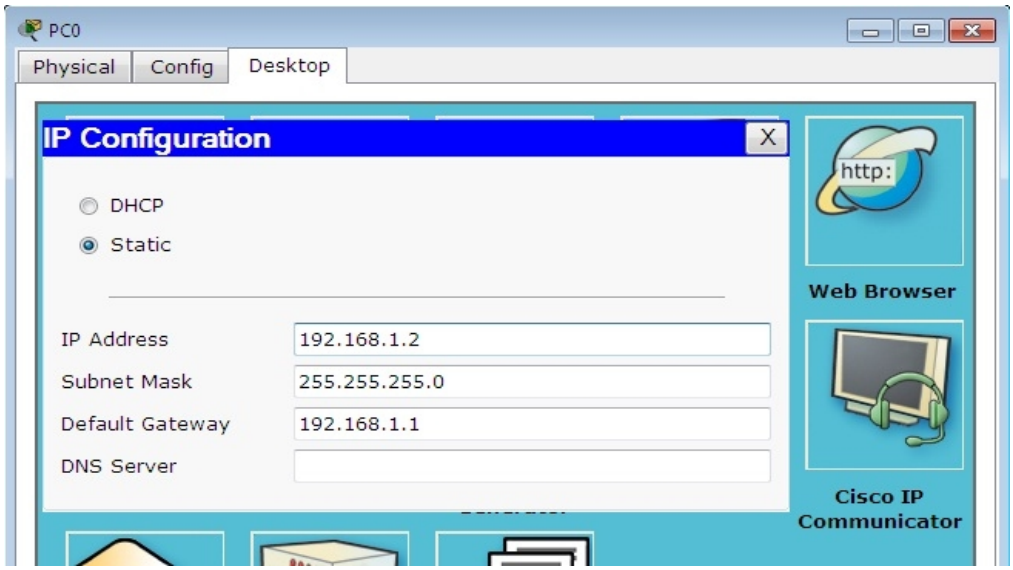

Figure-11: IP Configuration Window of PC0

Here the **IP address of PC-0 is:** 192.168.1.2
Default Gateway: 192.168.1.1 which is nothing but the one side (IP address of 0/0 port of router-0, towards PC-0) IP address of router-0

NOTE: Here PC-0 and 0/0 port of router-0 are in a same network that is why their network id is same(192.168.1)

IP Configuration of PC-1:

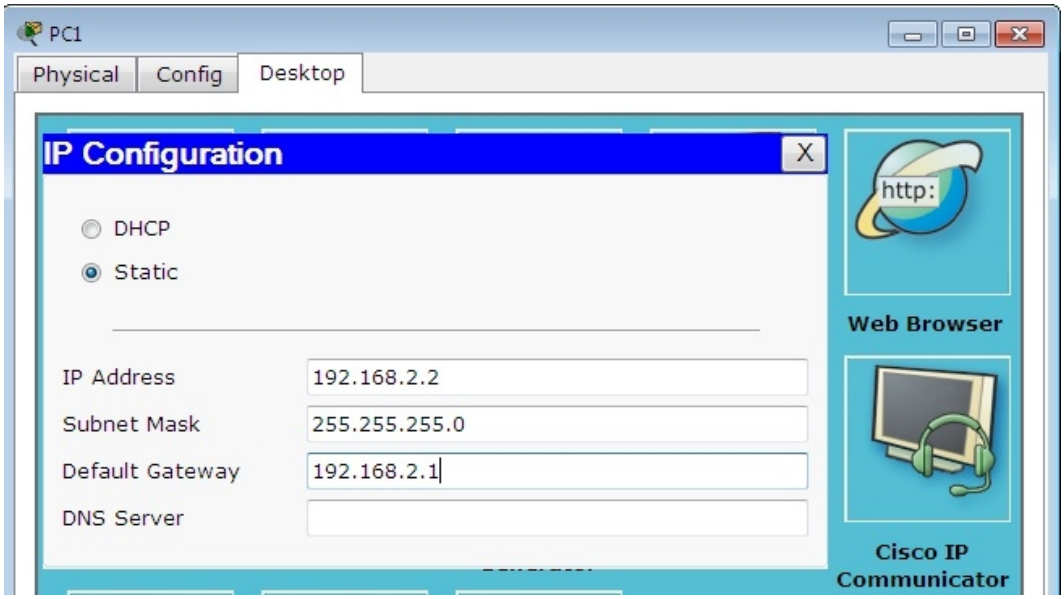

Figure-12: IP Configuration Window of PC1

Here the **IP address of PC-1 is:** 192.168.2.2
Default Gateway: 192.168.2.1 which is nothing but the one side (IP address of 1/0 port of router-0, towards PC-1) IP address of router-0

NOTE: Here PC-1 and 1/0 port of router-0 are in a same network that is why their network id is same (192.168.2)

Output:
After configuration the network looks like

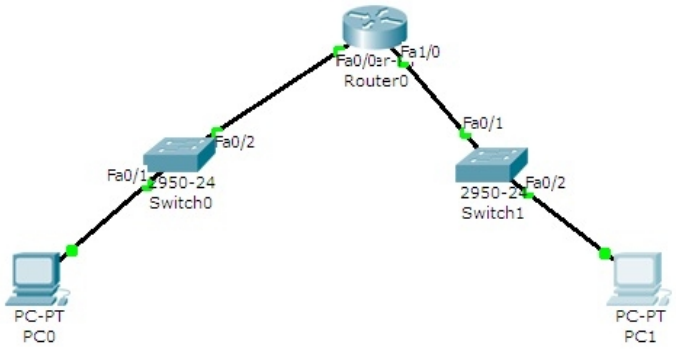

Figure-13: Network Topology

And we can ping PC-1 from PC-0 and If the connection is ok then then PC-0 will get reply from PC-1

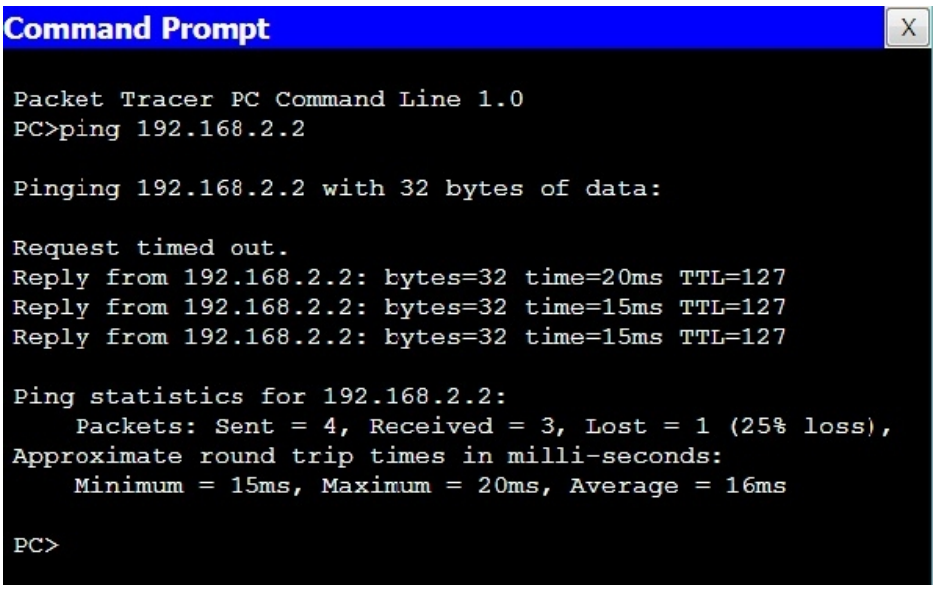

Figure-14: Output Screen

Experiment No-7 CONFIGURE NETWORK TOPOLOGY USING PACKET TRACER TO FIND THE ROUTING PATH BY IPROUTE COMMAND

Aim: To configure network topology to find the routing path.

Description: Routing is the process of selecting a path for traffic in a network, or between or across multiple networks. Routing is performed for many types of networks, including circuit-switched networks, such as the public switched telephone network (PSTN), computer networks, such as the Internet, as well as in networks used in public and private transportation, such as the system of streets, roads, and highways in national infrastructure.

Apparatus (Software): Packet Tracer 5.5.

Procedure: To implement this practical following network topology is required to be configured using the commands learned in previous experiment. After configuring the given network a packet should be ping from any one machine to another.

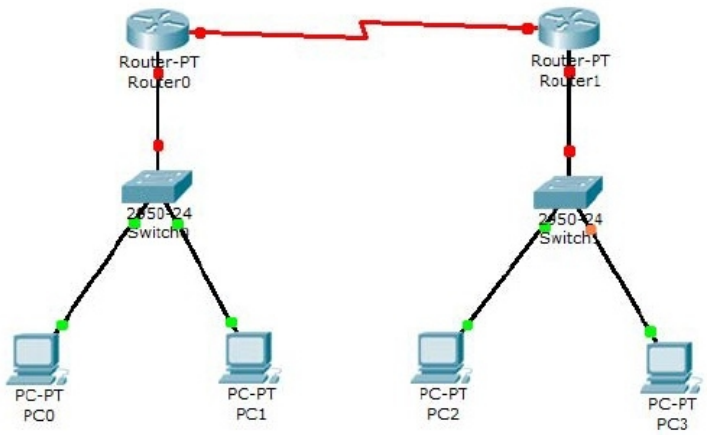

Figure-15: Network Topology.

Router-0 Configuration:

Router>
Router>Enable
Router>config t

```
Router(config)# hostname router0
router0(config)# interface fastethernet 0/0
router0(config-if)# ip address 192.168.0.254 255.255.255.0
router0(config-if)# no shutdown
router0(config-if)# exit
router0(config)# interface Serial2/0
router0(config-if)# ip address 192.168.1.1 255.255.255.0
router0(config-if)# no shutdown
router0(config-if)# exit
router0(config)# exit
router0#wr
Building configuration….
[OK]
router0# show running-config
```

< Student should write all comments/message to their laboratory note book after the execution of above command>

Router-1 Configuration:

```
Router> enable
Router#
Router# configure terminal
Router(config)# interface Serial2/0
Router(config-if)# ip address 192.168.1.2.255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# exit
Router# config t
Router(config-if)# ip address 192.168.2.254 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)#exit
Router(config)#exit
Router#wr
Building configuration…
[OK]
Router#
Router# show running-config
```

< Student should write all comments/message to their laboratory note book after the execution of above command>

IP ROUTE Command:

```
Router> enable
Router> show ip route
192.168.0.0/24[1/0] via 192.168.1.1
192.168.0.0/24 is directly connected, FastEthernet0/0
192.168.1.1/24 is directly connected, Serial2/0
```

Experiment No-8 NETWORK CONFIGURATION USING DISTANCE VECTOR ROUTING PROTOCOL

Aim: To configure a network using Distance Vector Routing Protocol. And the well known protocol is Routing Information Protocol (RIP)

Description: One of the most important examples of distance vector routing protocol is routing information protocol. The Routing Information Protocol (RIP) defines a way for routers, which connect networks using the Internet Protocol (IP), to share information about how to route traffic among networks. RIP is classified by the Internet Engineering Task Force (IETF) as an Interior Gateway Protocol (IGP), one of several protocols for routers moving traffic around within a larger autonomous system network -- e.g., a single enterprise's network that may be comprised of many separate local area networks (LANs) linked through routers.

Each RIP router maintains a routing table, which is a list of all the destinations (networks) it knows how to reach, along with the distance to that destination. RIP uses a distance vector algorithm to decide which path to put a packet on to get to its destination. It stores in its routing table the distance for each network it knows how to reach, along with the address of the "next hop" router -- another router that is on one of the same networks -- through which a packet has to travel to get to that destination. If it receives an update on a route, and the new path is shorter, it will update its table entry with the length and next-hop address of the shorter path; if the new path is longer, it will wait through a "hold-down" period to see if later updates reflect the higher value as well, and only update the table entry if the new, longer path is stable.

Apparatus (Software): Packet Tracer 5.5.

Procedure:

1. Develop a Topology shown in figure-16 given below
2. Configure all routers
3. Implement RIP protocol in Router in order to configure the network.

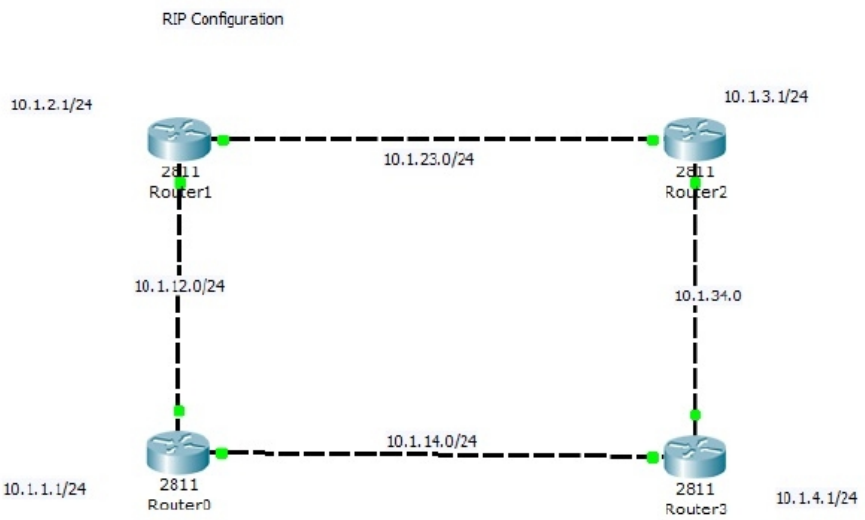

Figure-16: Network Topology.

Router-0 Configuration:

Router> en
Router> config t

```
router0(config)# int lo0
router0(config-if)#ip address 10.1.1.1 255.255.255.0
router0(config-if)#no shut
router0(config-if)#int f0/1
router0(config-if)#ip address 10.1.14.1 255.255.255.0
router0(config-if)#no shut
router0(config-if)#end
router0#wr
Building configuration…
[OK]
router0#
router0 con0 is now available
Press RETURN to get started.
router0>en
router0# config t
router0(config)# router rip
router0(config-router)#net 10.0.0.0
router0(config-router)#
router0(config-router)# end
router0#
```

Router-1 Configuration:

```
Router> en
Router> config t
Router(config)# hostname router1
router1 (config)# int lo0
router1 (config-if)#ip address 10.1.2.1 255.255.255.0
router1 (config-if)#no shut
router1 (config-if)#int f0/1
router1 (config-if)#ip address 10.1.23.1 255.255.255.0
router1 (config-if)#no shut
router1 (config-if)#int f0/0
router1 (config-if)# ip address 10.1.12.2 255.255.255.0
router1 (config-if)# no shut
router1 (config-if)#end
router1#wr
Building configuration…
[OK]
router1#
router1 con0 is now available
Press RETURN to get started.
router1>en
router1# config t
router1 (config)# router rip
router1 (config-router)#net 10.0.0.0
router1 (config-router)#
router1 (config-router)# end
router1#
```

Router-2 Configuration:

```
Router> en
Router> config t
Router(config)# hostname router2
```

```
router2 (config)# int lo0
router2 (config-if)#ip address 10.1.3.1      255.255.255.0
router2 (config-if)#no shut
router2 (config-if)#int f0/1
router2 (config-if)#ip address 10.1.34.1    255.255.255.0
router2 (config-if)#no shut
router2 (config-if)#int f0/0
router2 (config-if)# ip address 10.1.23.2     255.255.255.0
router2 (config-if)# no shut
router2 (config-if)#end
router2#wr
Building configuration…
[OK]
router2#
router2 con0 is now available
Press RETURN to get started.
router2>en
router2# config t
router2 (config)# router rip
router2 (config-router)#net 10.0.0.0
router2 (config-router)#
router2 (config-router)# end
router2#
```

Router-3 Configuration:

```
Router> en
Router> config t
Router(config)# hostname router3
router3 (config)# int lo0
router3 (config-if)#int f0/1
router3 (config-if)#ip address 10.1.14.2   255.255.255.0
router3 (config-if)#no shut
router3 (config-if)#int f0/0
router3 (config-if)# ip address 10.1.34.2     255.255.255.0
router3 (config-if)# no shut
router3 (config-if)#end
router3#wr
Building configuration…
[OK]
router3#
router3 con0 is now available
Press RETURN to get started.
router3>en
router3# config t
router3 (config)# router rip
router3 (config-router)#net 10.0.0.0
router3 (config-router)#
router3 (config-router)# end
router3#
```

Experiment No-9 CONFIGURATION OF DHCP PROTOCOL

Aim: To configure Dynamic Host Configuration Protocol in order to assign IP address to a PC or router dynamically.

Description: Dynamic Host Configuration Protocol (DHCP) is a protocol for assigning dynamic IP addresses to devices on a network. With dynamic addressing, a device can have a different IP address every time it connects to the network. In some systems, the device's IP address can even change while it is still connected. DHCP also supports a mix of static and dynamic IP addresses.

Apparatus (Software): Packet Tracer 5.5.

Procedure:

1. Develop a Topology shown in figure-17 given below
2. Configure DHCP in router0

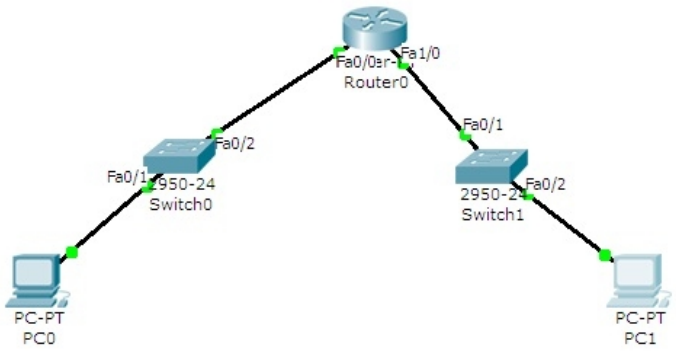

Figure-17: Network Topology.

Configuration of DHCP in Router-0

```
Router>en
Router# conf t
Router(config)# host R1
R1(config)# int fa1/0
R1(config-if)# ip add 192.168.10.1.2  255.255.255.0
R1(config-if)# no shutdown
R1(config-if)#exit
R1(config)# ip dhcp pool IP10  #pool of 10 Ip addresses #
R1(dhcp-config)#net 192.168.10.0 255.255.255.0
R1(dhcp-config)# default 192.168.10.1     #Router address#
R1(dhcp-config)#exit
R1(config)# ip dhcp exc 192.168.10.1 192.168.10.10
R1(config)# exit
R1#
```

Output from PC-1:

After the above configuration IP address of PC-1 will be assigned automatically after the request of IP address from the pool of IP address.

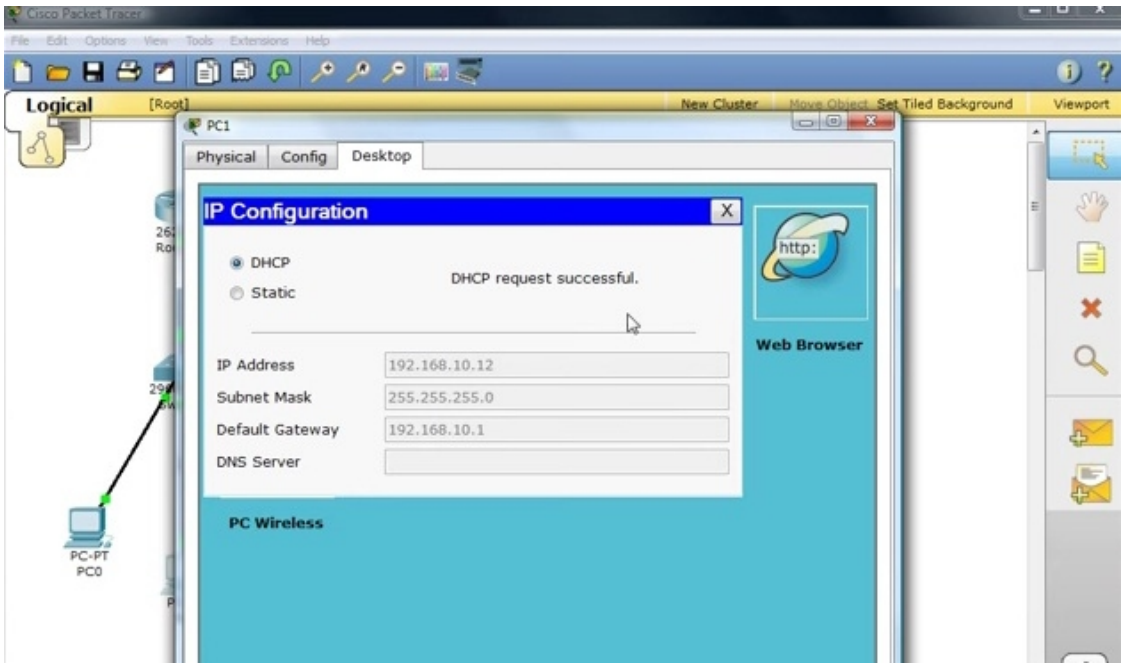

Figure-18: IP configuration box of PC1

Experiment No-10 TELNET CONFIGURATION

Aim: To configure TELNET Protocol in switch to get access of remote device

Description: Telnet is a terminal emulation program for TCP/IP networks such as the Internet. The Telnet program runs on your computer and connects your PC to a server on the network. You can then enter commands through the Telnet program and they will be executed as if you were entering them directly on the server console. This enables you to control the server and communicate with other servers on the network. To start a Telnet session, you must log in to a server by entering a valid username and password. Telnet is a common way to remotely control Web servers. The Telnet protocol is designed to provide a bi-directional, eight-bit byte oriented communications facility to allow for a standard method of interfacing terminal devices and processes. During Telnet configuration in switch we have to implement the concept of VLAN and its port.

About VLAN: A virtual LAN (VLAN) abstracts the idea of the LAN; A VLAN might comprise a subset of the ports on a single switch or subsets of ports on multiple switches. By default, systems on one VLAN don't see the traffic associated with systems on other VLANs on the same network. VLANs allow network administrators to partition their networks to match the functional and security requirements of their systems without having to run new cables or make major changes in their current network infrastructure. IEEE 802.1Q is the standard defining VLANs; the VLAN identifier or tag consists of 12 bits in the Ethernet frame, creating an inherent limit of 4,096 VLANs on a LAN. Ports on switches can be assigned to one or more VLANs, allowing systems to be divided into logical groups -- e.g., based on which department they are associated with -- and rules to be established about how systems in the separate groups are allowed to communicate with each other. These can range from the simple and practical (computers in one VLAN can see the printer on that VLAN, but computers outside that VLAN cannot), to the complex and legal (e.g., computers in the trading departments cannot interact with computers in the retail banking departments).

Apparatus (Software): Packet Tracer 5.5.

Procedure:

1. Develop a network (local area) shown in figure-19 given below
2. Configure TELNET in switch

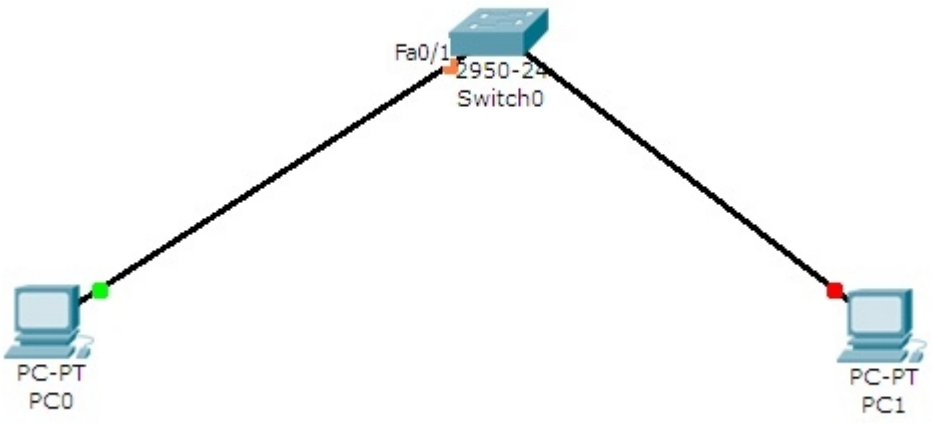

Figure-19: Network Topology

Switch Configuration:

```
Switch>
Switch>en
Switch# conf t
Switch(config)# line console 0
Switch(config-line)# pass cisco
Switch(config-line)# login
Switch(config-line)# exit
Switch(config)# line vty 0 4
Switch(config-line)# pass cisco
Switch(config-line)# login
Switch(config-line)# exit
Switch(config)# enable pass cisco
Switch(config)# int vlan 1
Switch(config-if)# ip add 172.17.1.1 255.255.0.0
Switch(config-if)# no shutdown
Switch(config-if)#exit
Switch(config)#exit
Switch# wr
Building Configuration...
```

Input & Output:

From PC-0 we can easily access remote switch by using TELNET.

Input: PC-0>ping 172.17.1.1

Figure-20: Output Screen of TELNET Configuration Test

Experiment No-11 CONFIGURATION OF ACCESS COMTROL LIST

Aim: To configure Access Control List (ACL) in order to give permission to the other IP addresses.

Description: An access control list (ACL), with respect to a computer file system, is a list of permissions attached to an object. An ACL specifies which users or system processes are granted access to objects, as well as what operations are allowed on given objects or you can say for each rule we have two conditions and that is Permit or Deny.

Types of Access Lists

There are two categories of access lists: numbered and named.

Numbered Access Lists:-

Numbered access lists are broken down into several ranges, each dedicated to a specific protocol:

- 1–99 IP standard access list
- 100-199 IP extended access list
- 200-299 Protocol type-code access list
- 300-399 DECnet access list
- 400-499 XNS standard access list
- 500-599 XNS extended access list
- 600-699 Appletalk access list
- 700-799 48-bit MAC address access list
- 800-899 IPX standard access list
- 900-999 IPX extended access list
- 1000-1099 IPX SAP access list
- 1100-1199 Extended 48-bit MAC address access list
- 1200-1299 IPX summary address access list
- 1300-1999 IP standard access list (expanded range)
- 2000-2699 IP extended access list

Named Access Lists:-

Named access lists provide a bit more flexibility. Descriptive names can be used to identify your access-lists. Additionally, individual lines can be removed from a named access-list. However, like numbered lists, all new entries are still added to the bottom of the access list.

There are two common types of named access lists:

• IP standard named access lists

• IP extended named access lists

Apparatus (Software): Packet Tracer 5.5.

Procedure:

Standard IP Access List:

Command Syntax: access-list [1-99] [permit | deny] [source address] [wildcard mask] [log]

Standard IP access-lists are based upon the source host or network IP address, and should be placed closest to the destination network.

Router(config)# access-list 10 deny 172.18.0.0 0.0.255.255 **(Just for an Example)**

Router(config)# access-list 10 permit any

To apply Access Lists we have to configure the Access-Group on the Interface. Likewise we are taking the interface serial 0 as a reference.

Router(config)# int s0

Router(config-if)# ip access-group 10 in

To view all IP access lists configured on the router:

Router# show ip access-list

To view what interface an access-list is configured on:

Router# show ip interface

Router# show running-config

Extended IP Access List:

Command Syntax: access-list [100-199] [permit | deny] [protocol] [source address] [wildcard mask] [destination address] [wildcard mask] [operator [port]] [log]

Router(config)# access-list 101 permit tcp 172.18.0.0 0.0.255.255 host 172.16.10.10 eq 80

Router(config)# access-list 101 deny ip 172.18.0.0 0.0.255.255 172.16.0.0 0.0.255.255

Router(config)# access-list 101 permit ip any any

NOTE: *The above IP address is just taken for the example and don't have real environment existence.

The first line allows the 172.18.x.x network access only to port 80 on the web server. The second line blocks 172.18.x.x from accessing anything else on the 172.16.x.x network. The third line allows 172.18.x.x access to anything else.

To apply this access list, we would configure the following

Router(config)# int e0

Router(config-if)# ip access-group 101 in

Extended IP Access List Port Operators

Router(config)# access-list 101 permit tcp 172.18.0.0 0.0.255.255 host 172.16.10.10 eq 80
We accomplished this using an operator of eq, which is short for equals. Thus, we are identifying host 172.16.10.10 with a port that equals 80. We can use several other operators for port numbers:

- **eq :-** Matches a specific port
- **gt :-** Matches all ports greater than the port specified
- **lt :-** Matches all ports less than the port specified
- **neq :-** Matches all ports except for the port specified
- **range:-** Match a specific inclusive range of ports

ICMP Access List
The specific ICMP port that a "ping" uses is echo. To block specific ICMP parameters, use anextended IP access list. On Router B, we would configure:

Router(config)# access-list 102 deny icmp 172.18.0.0 0.0.255.255 172.16.0.0 0.0.255.255 echo
Router(config)# access-list 102 permit icmp 172.18.0.0 0.0.255.255 172.16.0.0 0.0.255.255
Router(config)# access-list 102 permit ip any any

The first line blocks only ICMP echo requests (pings). The second line allows all other ICMP traffic. The third line allows all other IP traffic.
To apply the access lists on other router, you need to configure the following as:-
Router(config)# int e0
Router(config-if)# ip access-group 102 in