LABORATORY MANUAL

COMPUTER NETWORKS
BCS502

DO's and DON'Ts

DO's and DON'Ts

Do's

Students should be in proper uniform and dress code with identity cards in the laboratory. Students should bring their observation, manual and record compulsorily.

Students should maintain discipline in the laboratory.

Students are required to handle all the equipment's/Computers properly.

Students are required to follow the safety precautions.

Enter the lab in time as per the given time table.

Enter time-in and time-out in log book.

Comply with the instructions given by faculty and instructor.

Arrange the chairs/ equipment's before leaving the lab.

Take signature in the observation, before leaving the lab.

Don'ts

Mobile phones are strictly banned.

Ragging is punishable.

Do not turn on the power supply before verification of the circuits by the Batch in Charge. Do not operate any peripherals or accessories without supervision.

Avoid stepping on computer cables and electrical wires. Do not walk around in the lab unnecessarily.

Do not go out of the lab without permission.

Institute Vision and Mission

1. Institute Vision and Mission

Institute Vision

Our vision is to provide learning opportunities, ensuring excellence in education, research and facilitate an inspiring world class environment to encourage creativity. The Institute is committed to disseminating knowledge, and through its ingenuity, bring this knowledge to bear on the world's great challenges. VVIET is dedicated to providing its students with an education that combines academic study and the excitement of discovery kindled by a diverse campus community.

Institute Mission

- Offer highest professional and academic standards in terms of personal growth and satisfaction, and promote growth and value to our research sponsors.
- Provide students a platform where independent learning and scientific study are encouraged with emphasis on latest engineering techniques.
- Encourage students to implement applications of engineering with a focus on societal needs for the betterment of communities.
- Empower students with vast technical and life skills to raise their stakes of getting placements in top reputed companies.
- Create a benchmark in the areas of research, education and public outreach.

Department Vision and Mission

2. Department Vision and Mission

Department Vision

"To produce proficient computer professionals, having essential technical knowledge and skills, with good work ethics"

Department Mission

M1: To promote growth of an individual by imparting comprehensive knowledge using latest tools and technologies.

M2: To inculcate professionalism, social awareness and to promote creativity, research aptitude by mentoring the students.

M3: To establish industry institute interaction, to enable students to cater the ever changing industry demands and to nurture entrepreneurial qualities.

M4: To provide state-of-the-art environment and opportunities to enhance professional skills.

Programme Educational Objectives

3. Programme Educational Objectives

PEO 1: To provide Graduates of computer science & engineering course with a solid foundation in the principles and practices of computer science and engineering enabling them to have successful professional career.

PEO 2: To encourage Graduates of computer science & engineering course to pursue higher education.

PEO 3: To prepare Graduates of computer science & engineering course to adapt to technological advancements by engaging in lifelong learning.

Programme Outcome

4. Program Outcomes (POs)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, research literature, and Analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Programme Specific Outcomes

5. Programme Specific Outcomes

PSO 1: Use the knowledge of algorithms and programming skills to efficiently build, test and maintain software systems.

PSO 2: Design and build systems, catering the needs of industry and society.

Specification of the Laboratory

SI.NO	MAJOR EQUPIMENT/SYSTEM	SPECIFICATION
1	Computer	CORE I5 8GB RAM 160GB HDD 17inch MONITOR
2	SOFTWARE	Ubuntu, NS2

Syllabus

Syllabus

COMPUTER NETWORKS (BCS502)

1. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.
2. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
4. Develop a program for error detecting code using CRC-CCITT (16- bits).
5. Develop a program to implement a sliding window protocol in the data link layer.
6. Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.
7. Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.
8. Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.
9. Develop a program for a simple RSA algorithm to encrypt and decrypt the data.
10. Develop a program for congestion control using a leaky bucket algorithm

Syeda Arbeena Kausar

Signature of the Lab In-charge **Signature of the HOD**

1. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.

tcl file:

```
set ns [ new Simulator ]
set tf [ open prog1.tr w ]
$ns trace-all $tf
set nf [ open prog1.nam w ]
$ns namtrace-all $nf
# The below code is used to create the nodes.
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#This is used to give color to the packets.
$ns color 1 "red"
$ns color 2 "blue"
$n0 label "Source/udp0"
$n1 label "Source/udp1"
$n2 label "Router"
$n3 label "Destination/Null"
#Vary the below Bandwidth and see the number of packetsdropped.
$ns duplex-link $n0 $n2 100Mb 300ms DropTail
$ns duplex-link $n1 $n2 10Mb 300ms DropTail
$ns duplex-link $n2 $n3 1Mb 300ms DropTail
#The below code is used to set the queue size b/w the nodes
$ns set queue-limit $n0 $n2 1
$ns set queue-limit $n1 $n2 1
$ns set queue-limit $n2 $n3 1
#The below code is used to attach an UDP agent to n0, UDP
#agent to n1 and null agent to n3.
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
set null3 [new Agent/Null]
$ns attach-agent $n3 $null3
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
#The below code sets the udp0 packets to red and udp1
#packets to blue color
$udp0 set class_ 1
```

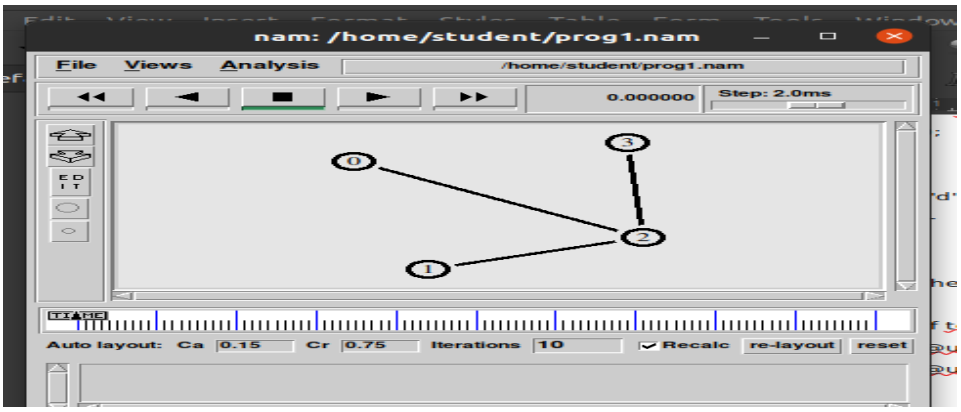
```
$udp1 set class_ 2
#The below code is used to connect the agents.
$ns connect $udp0 $null3
$ns connect $udp1 $null3
#The below code is used to set the packet size to 500
$cbr1 set packetSize_ 500Mb
#The below code is used to set the interval of the packets,
#i.e., Data rate of the packets. if the data rate is high
#then packets drops are high.
$cbr1 set interval_ 0.005
proc finish { } {
global ns nftf
$ns flush-trace
exec nam prog1.nam &
close $tf
close $nf
exit 0
}
$ns at 0.1 "$cbr0 start"
$ns at 0.1 "$cbr1 start"
$ns at 10.0 "finish"
$ns run
```

awk file :

```
BEGIN{
#include<stdio.h>
count=0;
}
{
if($1=="d") #d stands for the packets drops.
count++
}
END{
printf("The Total no of Packets Dropped due to Congestion :%d\n\n", count)
}
```

output of tcl file :

student@user-ThinkCentre-neo-50t-Gen-3:~$ gedit prog1.tcl
student@user-ThinkCentre-neo-50t-Gen-3:~$ ns prog1.tcl



Output of awk file :

student@user-ThinkCentre-neo-50t-Gen-3:~$ gedit prog1.awk
student@user-ThinkCentre-neo-50t-Gen-3:~$ awk -f prog1.awk prog1.tr
The Total no of Packets Dropped due to Congestion :829

2. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

```
#Create Simulator
set ns [new Simulator]
#Use colors to differentiate the traffic
$ns color 1 Blue
$ns color 2 Red
#Open trace and NAM trace file
set ntrace [open pg2.tr w]
$ns trace-all $ntrace
set namfile [open pg2.nam w]
$ns namtrace-all $namfile
#Finish Procedure
proc Finish {} {
global ns ntrace namfile
#Dump all trace data and close the file
$ns flush-trace
close $ntrace
close $namfile
#Execute the nam animation file
exec nam prog3.nam &
#Find the number of ping packets dropped
puts "The number of ping packets dropped are "
exec grep "^d" pg2.tr | cut -d " " -f 5 | grep -c "ping" &
exit 0
}
#Create six nodes
for {set i 0} {$i < 6} {incr i} {
set n($i) [$ns node]
}
#Connect the nodes
for {set j 0} {$j < 5} {incr j} {
$ns duplex-link $n($j) $n([expr ($j+1)]) 0.1Mb 10ms DropTail
}
#Define the recv function for the class 'Agent/Ping'
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] received ping answer from $from with round trip time $rtt
ms"
}
#Create two ping agents and attach them to n(0) and n(5)
set p0 [new Agent/Ping]
$p0 set class_ 1
```

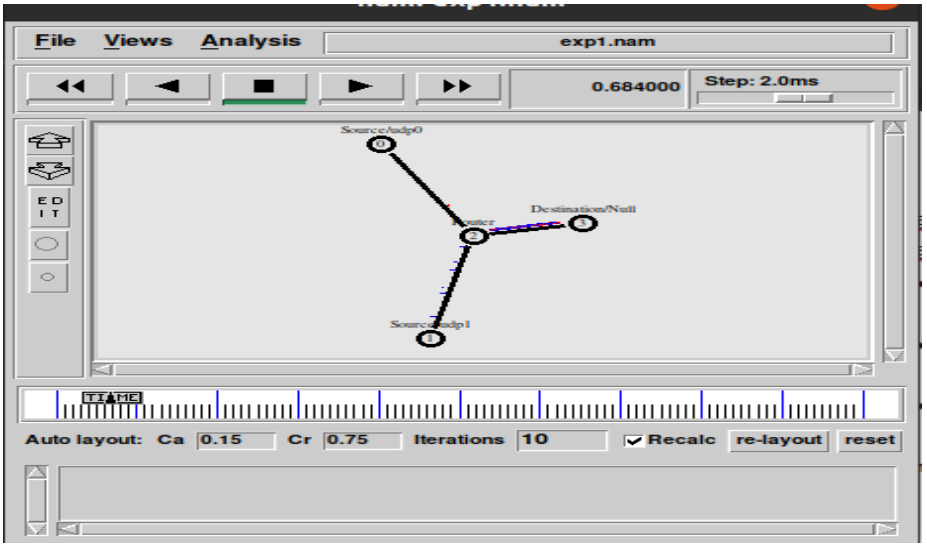
```
$ns attach-agent $n(0) $p0
set p1 [new Agent/Ping]
$p1 set class_ 1
$ns attach-agent $n(5) $p1
$ns connect $p0 $p1

#Set queue size and monitor the queue
#Queue size is set to 2 to observe the drop in ping packets
$ns queue-limit $n(2) $n(3) 2
$ns duplex-link-op $n(2) $n(3) queuePos 0.5
#Create Congestion
#Generate a Huge CBR traffic between n(2) and n(4)
set tcp0 [new Agent/TCP]
$tcp0 set class_ 2
$ns attach-agent $n(2) $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0
#Apply CBR traffic over TCP
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set rate_ 1Mb
$cbr0 attach-agent $tcp0
#Schedule events
$ns at 0.2 "$p0 send"
$ns at 0.4 "$p1 send"
$ns at 0.4 "$cbr0 start"
$ns at 0.8 "$p0 send"
$ns at 1.0 "$p1 send"
$ns at 1.2 "$cbr0 stop"
$ns at 1.4 "$p0 send"
$ns at 1.6 "$p1 send"
$ns at 1.8 "Finish"
#Run the Simulation
$ns run
```

output

```
student@user-ThinkCentre-neo-50t-Gen-3:~$ gedit pg2.tcl
student@user-ThinkCentre-neo-50t-Gen-3:~$ ns pg2.tcl
node 0 received ping answer from 5 with round trip time 151.2
ms
node 0 received ping answer from 5 with round trip time 301.4
ms
```

node 5 received ping answer from 0 with round trip time 155.4
ms
The number of ping packets dropped are
3



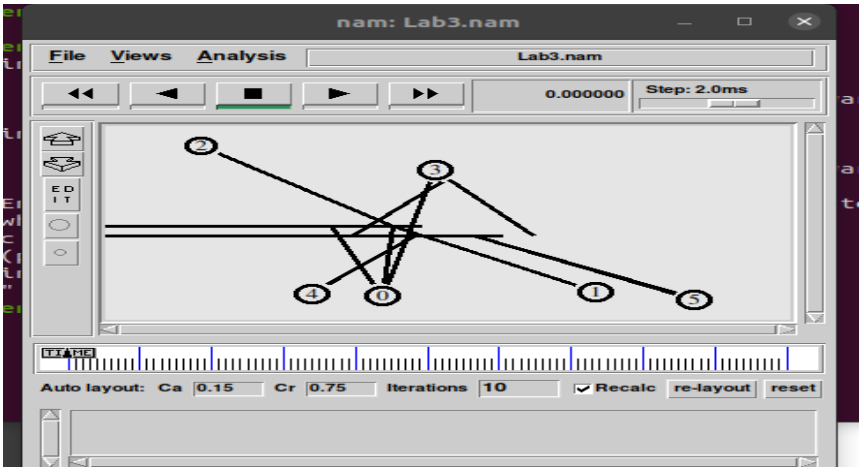
3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

```
set ns [new Simulator]
$ns color 1 Red
$ns color 2 Blue
set na [open Lab3.nam w]
$ns namtrace-all $na
set nt [open Lab3.tr w]
$ns trace-all $nt
set ng1 [open tcp1.xg w]
set ng2 [open tcp2.xg w]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns make-lan "$n0 $n1 $n2" 1Mb 10ms LL Queue/DropTail Mac/802_3
$ns make-lan "$n3 $n4 $n5" 2Mb 10ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
set tcp1 [new Agent/TCP]
set tcp2 [new Agent/TCP]
set cbr1 [new Application/Traffic/CBR]
set cbr2 [new Application/Traffic/CBR]
$ns attach-agent $n4 $tcp1
$cbr1 attach-agent $tcp1
$ns attach-agent $n1 $tcp2
$cbr2 attach-agent $tcp2
set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $sink1
$ns attach-agent $n5 $sink2
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
proc End {} {
global ns na nt
$ns flush-trace
close $na
close $nt
exec nam Lab3.nam &
exec xgraph tcp1.xg tcp2.xg &
exit 0
```

```
}
proc Draw {Agent File} {
global ns
set Cong [$Agent set cwnd_]
set Time [$ns now]
puts $File "$Time $Cong"
$ns at [expr $Time+0.01] "Draw $Agent $File"
}
$ns at 0.0 "$cbr1 start"
$ns at 0.7 "$cbr2 start"
$ns at 0.0 "Draw $tcp1 $ng1"
$ns at 0.0 "Draw $tcp2 $ng2"
$ns at 10.0 "End"
$ns run
```

Output :



4. Write a program for error detecting code using CRC-CCITT (16- bits).

```
import java.util.Scanner;

public class CRC_CCITT_Custom {

    // Method to perform CRC computation
    public static String computeCRC(String data, String divisor) {
        int dataLength = data.length();
        int divisorLength = divisor.length();

        // Append zero bits to the end of the data, based on divisor length
        String paddedData = data + "0".repeat(divisorLength - 1);

        // Convert data and divisor to arrays for easy manipulation
        char[] dataBits = paddedData.toCharArray();
        char[] divisorBits = divisor.toCharArray();

        // Perform bitwise division
        for (int i = 0; i <= dataLength - 1; i++) {
            if (dataBits[i] == '1') {
                // Perform XOR with the divisor bits
                for (int j = 0; j < divisorLength; j++) {
                    dataBits[i + j] = dataBits[i + j] == divisorBits[j] ? '0' : '1';
                }
            }
        }

        // Extract remainder as the last part of dataBits
        StringBuilder remainder = new StringBuilder();
        for (int k = dataLength; k < dataBits.length; k++) {
            remainder.append(dataBits[k]);
        }

        return remainder.toString();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Get data size from user
        System.out.println("Enter the size of data (in bits): ");
        int dataSize = scanner.nextInt();
```

```
        scanner.nextLine(); // Consume newline

        // Get binary data from user
        System.out.println("Enter the binary data: ");
        String data = scanner.nextLine();

        // Validate data length
        if (data.length() != dataSize) {
            System.out.println("Error: Data length does not match the specified size.");
            return;
        }

        // Get divisor from user
        System.out.println("Enter the divisor (polynomial) in binary: ");
        String divisor = scanner.nextLine();

        // Calculate CRC using the given divisor
        String crcRemainder = computeCRC(data, divisor);

        // Display the result
        System.out.println("Computed CRC remainder: " + crcRemainder);
        System.out.println("Transmitted data with CRC remainder: " + data + crcRemainder);

        scanner.close();
    }
}
```

OUTPUT

```
Enter the size of data (in bits):
5
Enter the binary data:
11100
Enter the divisor (polynomial) in binary:
1001
Computed CRC remainder: 111
Transmitted data with CRC remainder: 11100111

=== Code Execution Successful ===
```

5. Develop a program to implement a sliding window protocol in the data link layer.

```
import java.util.Scanner;

class SlidingWindowProtocol {
    private int windowSize;
    private int[] frameStatus;
    private int totalFrames;

    public SlidingWindowProtocol(int totalFrames, int windowSize) {
        this.totalFrames = totalFrames;
        this.windowSize = windowSize;
        this.frameStatus = new int[totalFrames];
    }

    public void simulateTransmission() {
        int sentFrames = 0;
        Scanner sc = new Scanner(System.in);

        while (sentFrames < totalFrames) {
            int framesToSend = Math.min(windowSize, totalFrames - sentFrames);

            System.out.println("\nSending frames: ");
            for (int i = sentFrames; i < sentFrames + framesToSend; i++) {
                System.out.println("Frame " + (i + 1) + " sent.");
                frameStatus[i] = 1;  // Mark as sent
            }

            // Simulate Acknowledgments
            for (int i = sentFrames; i < sentFrames + framesToSend; i++) {
                System.out.print("Is frame " + (i + 1) + " acknowledged? (y/n): ");
                char ack = sc.next().charAt(0);

                if (ack == 'y' || ack == 'Y') {
                    System.out.println("Frame " + (i + 1) + " acknowledged.");
                    frameStatus[i] = 2;  // Mark as acknowledged
                } else {
                    System.out.println("Frame " + (i + 1) + " not acknowledged. Resending from frame "
+ (i + 1) + ".");
                    sentFrames = i;  // Go back to this frame
                    break;
                }

                if (i == sentFrames + framesToSend - 1) {
```

```
                sentFrames += framesToSend;  // Move the window forward if all frames are
                acknowledged
            }
        }
    }

    System.out.println("\nAll frames have been transmitted successfully!");
    sc.close();
  }
}

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print("Enter the total number of frames: ");
    int totalFrames = sc.nextInt();

    System.out.print("Enter the window size: ");
    int windowSize = sc.nextInt();

    SlidingWindowProtocol swp = new SlidingWindowProtocol(totalFrames, windowSize);
    swp.simulateTransmission();

    sc.close();
  }
}
```

OUTPUT :-

```
Enter the total number of frames: 5
Enter the window size: 3

Sending frames:
Frame 1 sent.
Frame 2 sent.
Frame 3 sent.
Is frame 1 acknowledged? (y/n): y
Frame 1 acknowledged.
Is frame 2 acknowledged? (y/n): y
Frame 2 acknowledged.
Is frame 3 acknowledged? (y/n): y
Frame 3 acknowledged.
Is frame 4 acknowledged? (y/n): y
```

Frame 4 acknowledged.
Is frame 5 acknowledged? (y/n): y
Frame 5 acknowledged.
Is frame 6 acknowledged? (y/n): y
Frame 6 acknowledged.
ERROR!
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
    at SlidingWindowProtocol.simulateTransmission(Main.java:34)
    at Main.main(Main.java:63)

6. Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.

```
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

// Class representing an edge between two vertices
class Edge {
    int source, destination, weight;

    Edge(int source, int destination, int weight) {
        this.source = source;
        this.destination = destination;
        this.weight = weight;
    }
}

public class BellmanFordPathVector {
    static final int INF = Integer.MAX_VALUE;

    // Bellman-Ford algorithm to find shortest path from the source to all vertices
    public static void bellmanFord(Edge[] edges, int V, int E, int src) {
        // Initialize distance array with infinity for all vertices except the source
        int[] distance = new int[V];
        Arrays.fill(distance, INF);
        distance[src] = 0;

        // Relax all edges V-1 times
        for (int i = 0; i < V - 1; i++) {
            for (int j = 0; j < E; j++) {
                int u = edges[j].source;
                int v = edges[j].destination;
                int weight = edges[j].weight;

                if (distance[u] != INF && distance[u] + weight < distance[v]) {
                    distance[v] = distance[u] + weight;
                }
            }
        }

        // Check for negative-weight cycles
        for (int j = 0; j < E; j++) {
            int u = edges[j].source;
```

```
            int v = edges[j].destination;
            int weight = edges[j].weight;

            if (distance[u] != INF && distance[u] + weight < distance[v]) {
                System.out.println("Graph contains a negative-weight cycle.");
                return;
            }
        }

        // Print the shortest distances
        printShortestPaths(distance, V);
    }

    // Path Vector Routing Algorithm: Determines the best route based on path vector
    public static void pathVectorRouting(Edge[] edges, int V, int E, int src) {
        Map<Integer, String> routingTable = new HashMap<>();
        for (int i = 0; i < V; i++) {
            routingTable.put(i, "Direct path");
        }

        // Update routing table with path information
        for (int i = 0; i < E; i++) {
            int u = edges[i].source;
            int v = edges[i].destination;
            routingTable.put(v, routingTable.get(u) + " -> " + v);
        }

        System.out.println("\nRouting Table (Path Vector):");
        for (int i = 0; i < V; i++) {
            System.out.println("Node " + i + ": " + routingTable.get(i));
        }
    }

    // Utility function to print the shortest paths
    private static void printShortestPaths(int[] distance, int V) {
        System.out.println("Vertex \t\t Distance from Source");
        for (int i = 0; i < V; i++) {
            if (distance[i] == INF) {
                System.out.println(i + " \t\t " + "INF");
            } else {
                System.out.println(i + " \t\t " + distance[i]);
            }
        }
```

```
    }
    public static void main(String[] args) {
        int V = 5;  // Number of vertices
        int E = 8;  // Number of edges

        // Define graph edges (source, destination, weight)
        Edge[] edges = new Edge[E];
        edges[0] = new Edge(0, 1, -1);
        edges[1] = new Edge(0, 2, 4);
        edges[2] = new Edge(1, 2, 3);
        edges[3] = new Edge(1, 3, 2);
        edges[4] = new Edge(1, 4, 2);
        edges[5] = new Edge(3, 2, 5);
        edges[6] = new Edge(3, 1, 1);
        edges[7] = new Edge(4, 3, -3);

        int src = 0;  // Source vertex

        // Bellman-Ford Algorithm to find the shortest paths
        System.out.println("Bellman-Ford Algorithm:");
        bellmanFord(edges, V, E, src);

        // Path Vector Routing Algorithm to find routes
        System.out.println("\nPath Vector Routing Algorithm:");
        pathVectorRouting(edges, V, E, src);
    }
}
```

OUTPUT:-

```
Enter the number of vertices: 5
Enter the adjacency matrix:
0 3 0 0 0
0 0 2 0 0
0 0 0 1 0
0 0 0 0 5
0 0 0 0 0
Enter the source vertex: 1
Shortest distance from source 1 to vertex 1: 0
Shortest distance from source 1 to vertex 2: 3
Shortest distance from source 1 to vertex 3: 5
Shortest distance from source 1 to vertex 4: 6
Shortest distance from source 1 to vertex 5: 11
```

7. Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.

server :

```
import java.net.*;
import java.io.*;
public class server {
   public static void main(String[] args) throws IOException {
      ServerSocket serverSocket = new ServerSocket(4000);
      System.out.println("Server listening on port 4000...");
      while (true) {
         Socket socket = serverSocket.accept();
         System.out.println("Client connected: " + socket.getInetAddress());
         try {
            InputStream inputStream = socket.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream));
            String filename = bufferedReader.readLine();
            System.out.println("Client requested file: " + filename);
            File file = new File(filename);
            if (!file.exists() || !file.isFile()) {
               System.out.println("File not found or not a file: " + filename);
               PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
               out.println("ERROR: File not found or not a file");
               out.close();
               continue;
            }

            FileInputStream fis = new FileInputStream(file);
            OutputStream outputStream = socket.getOutputStream();
            byte[] buffer = new byte[4096];
            int bytesRead;
            while ((bytesRead = fis.read(buffer)) != -1) {
               outputStream.write(buffer, 0, bytesRead);
```

```
            }
            System.out.println("File sent successfully.");
            fis.close();
        outputStream.close();
        } catch (IOException e) {
                System.err.println("Error: " + e.getMessage());
        } finally {
            socket.close();
        }
    }
  }
}
```

client :

```
import java.net.*;
import java.io.*;
import java.util.Scanner;
public class client {
   public static void main(String[] args) throws IOException {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the filename: ");
      String filename = scanner.nextLine();
      Socket socket = new Socket("localhost", 4000);
      System.out.println("Connected to server.");
      OutputStream outputStream = socket.getOutputStream();
      PrintWriter out = new PrintWriter(outputStream, true);
      out.println(filename);
      InputStream inputStream = socket.getInputStream();
      BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream));
      String response = bufferedReader.readLine();
      if (response.equals("ERROR: File not found or not a file")) {
         System.out.println("File not found or not a file on the server.");
      } else {
```

```
            System.out.println("Receiving file...");
            FileOutputStream fos = new FileOutputStream(filename);
            byte[] buffer = new byte[4096];
            int bytesRead;
            while ((bytesRead = inputStream.read(buffer)) != -1) {
                fos.write(buffer, 0, bytesRead);
            }
            System.out.println("File received successfully.");
            fos.close();
        }
        out.close();
        inputStream.close();
        socket.close();
    }
}
```

OUTPUT:-

Server output

```
student@user-ThinkCentre-neo-50t-Gen-3:~$ gedit server.java
student@user-ThinkCentre-neo-50t-Gen-3:~$ java server
Server listening on port 4000...
Client connected: /127.0.0.1
Client requested file: client.java
File sent successfully.
```

Client output:

```
student@user-ThinkCentre-neo-50t-Gen-3:~$ gedit client.java
student@user-ThinkCentre-neo-50t-Gen-3:~$ java client
Enter the filename: client.java
Connected to server.
Receiving file...
File received successfully.
```

8. Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.

Server

```
import java.net.*;
import java.io.*;

public class udpserver {
   public static void main(String[] args) throws Exception {
      int port = 3333; // Replace with your desired port number

      DatagramSocket serverSocket = new DatagramSocket(port);
      System.out.println("Server started on port " + port);

      byte[] receiveBuffer = new byte[1024];
      byte[] sendBuffer;

      while (true) {
         DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
         serverSocket.receive(receivePacket);

         String message = new String(receivePacket.getData(), 0, receivePacket.getLength());
         System.out.println("Client: " + message);


         String ack = "Message received";
         sendBuffer = ack.getBytes();
         DatagramPacket ackPacket = new DatagramPacket(sendBuffer, sendBuffer.length, receivePacket.getAddress(), receivePacket.getPort());
         serverSocket.send(ackPacket);

         if (message.equalsIgnoreCase("stop")) {
            break;
         }
      }

      serverSocket.close();
      System.out.println("Server stopped.");
   }
}
```

Client:

```
import java.net.*;
import java.io.*;

public class udpclient {
   public static void main(String[] args) throws Exception {
      int port = 3333;
      String serverAddress = "localhost";

      DatagramSocket clientSocket = new DatagramSocket();

      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      String message;

      while (true) {
         message = br.readLine();

         byte[] sendBuffer = message.getBytes();
         DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length,
InetAddress.getByName(serverAddress), port);
         clientSocket.send(sendPacket);

         if (message.equalsIgnoreCase("stop")) {
            break;
         }

          byte[] receiveBuffer = new byte[1024];
         DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);
         clientSocket.receive(receivePacket);
         String ack = new String(receivePacket.getData(), 0, receivePacket.getLength());
         System.out.println("Server: " + ack);
      }

      clientSocket.close();
      System.out.println("Client stopped.");
   }
}
```

OUTPUT:-

Server

```
student@user-ThinkCentre-neo-50t-Gen-3:~$ gedit udpserver.java
student@user-ThinkCentre-neo-50t-Gen-3:~$ javac udpserver.java
student@user-ThinkCentre-neo-50t-Gen-3:~$ java udpserver
Server started on port 3333
Client: vviet
```

Client

```
student@user-ThinkCentre-neo-50t-Gen-3:~$ gedit client.java
student@user-ThinkCentre-neo-50t-Gen-3:~$ java client
Enter the filename: client.java
Connected to server.
Receiving file...
File received successfully.
```

9. Develop a program for a simple RSA algorithm to encrypt and decrypt the data.

```
import java.math.BigInteger;
import java.security.SecureRandom;
import java.util.Scanner;

public class RSA {
   private BigInteger p, q, N, e, d;

   public RSA() {
      generateKeys();
   }

   private void generateKeys() {
      SecureRandom random = new SecureRandom();
      p = BigInteger.probablePrime(512, random);
      q = BigInteger.probablePrime(512, random);
      N = p.multiply(q);
      BigInteger phiN = (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

      e = BigInteger.probablePrime(512, random);
      while (e.gcd(phiN).compareTo(BigInteger.ONE) > 0) {
         e = e.add(BigInteger.ONE);
      }

      d = e.modInverse(phiN);
   }

   public BigInteger encrypt(BigInteger message) {
      return message.modPow(e, N);
   }

   public BigInteger decrypt(BigInteger ciphertext) {
      return ciphertext.modPow(d, N);
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the message to encrypt: ");
      String message = scanner.nextLine();

      RSA rsa = new RSA();
```

```
        BigInteger plaintext = new BigInteger(message.getBytes());
        BigInteger ciphertext = rsa.encrypt(plaintext);
        BigInteger decryptedText = rsa.decrypt(ciphertext);

        System.out.println("Encrypted message: " + ciphertext);
        System.out.println("Decrypted message: " + new String(decryptedText.toByteArray()));
    }
}
```

OUTPUT:-

```
Enter the message to encrypt: Hello welcome to "Computer Networks lab"!!!!
Encrypted message:
311790378939717630959340753217562448742111649935663650843676358972353836021
525146068996770350924783696011186145639224648855022947609921437973463040740
886007315112902850121782024405281349909090542750621722589551089616305606052
421629577197034824414878855082510115733984098745439398537631450452466497289
99096887
Decrypted message: Hello welcome to "Computer Networks lab"!!!!
```

10. Develop a program for congestion control using a leaky bucket algorithm.

```
import java.util.Timer;
import java.util.TimerTask;
public class LeakyBucketAlgorithm {
    private static final int BUCKET_SIZE = 10; // Maximum tokens in the bucket
    private static final int TOKEN_RATE = 2; // Tokens added per second
    private int currentTokens;
    private Timer timer;
    public LeakyBucketAlgorithm() {
        currentTokens = BUCKET_SIZE;
        timer = new Timer();
        timer.scheduleAtFixedRate(new TokenRefillTask(), 0, 1000 / TOKEN_RATE);
    }
    public synchronized boolean sendPacket() {
        if (currentTokens > 0) {
            currentTokens--;
            return true;
        } else {
            return false;
        }
    }
    private class TokenRefillTask extends TimerTask {
        @Override
        public void run() {
            if (currentTokens < BUCKET_SIZE) {
                currentTokens++;
            }
        }
    }
    public static void main(String[] args) {
        LeakyBucketAlgorithm leakyBucket = new LeakyBucketAlgorithm();
        // Simulate sending packets
        for (int i = 0; i < 20; i++) {
            if (leakyBucket.sendPacket()) {
                System.out.println("Packet sent: " + i);
            } else {
                System.out.println("Packet dropped: " + i);
            }
            try {
                Thread.sleep(100); // Simulate packet transmission time
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
```

```
    }
  }
}
```

OUTPUT:-This output shows that the leaky bucket algorithm effectively controls the rate of packet transmission, dropping packets when the bucket is full to prevent congestion.

```
Packet sent: 0
Packet sent: 1
Packet sent: 2
Packet sent: 3
Packet sent: 4
Packet sent: 5
Packet sent: 6
Packet sent: 7
Packet sent: 8
Packet sent: 9
Packet sent: 10
Packet sent: 11
Packet sent: 12
Packet dropped: 13
Packet dropped: 14
Packet sent: 15
Packet dropped: 16
Packet dropped: 17
Packet dropped: 18
Packet dropped: 19
```