**KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY**

**(AN AUTONOMOUS INSTITUTION)**

### Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH,

### Narayanguda, Hyderabad – 500029





**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**LAB RECORD**

**COMPUTER NETWORKS LAB**

**B.Tech. III YEAR I SEM (KR21)**

**ACADEMIC YEAR 2023-24**

****

**KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY**

**(AN AUTONOMOUS INSTITUTE)**

**Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Hyderabad**

**Department of Computer Science & Engineering**

**Certificate**

This is to certify that following is a Bonafide Record of the workbook task done by \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_bearing Roll No\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ of \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Branch of \_\_\_\_\_\_\_\_\_\_\_ year B.Tech Course in the\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Subject during the Academic year \_\_\_\_\_\_\_\_\_\_\_\_ & \_\_\_\_\_\_\_\_\_\_ under our supervision.

Number of week tasks completed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature of Staff Member Incharge Signature of Head of the Dept.

****

**KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY**

**(AN AUTONOMOUS INSTITUTE)**

**Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Hyderabad**

**Department of Computer Science & Engineering**

**Daily Laboratory Assessment Sheet**

Name of the Lab: Name of the Student:

Class: HT.No:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | **Name of the Experiment** | **Date** | **Observation Marks (3M)** | **Record Marks (4M)** | **Viva Voice Marks (3M)** | **Total Marks (10M)** | **Signature of Faculty** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  | **TOTAL** |  |  |  |  |  |  |

**Faculty Incharge**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE**  **NO** |
| I | Vision/Mission /PEOs/POs/PSOs | 6 |
| II | Syllabus | 12 |
| III | Course outcomes, CO-PO Mapping | 14 |
| **Exp No** | **List of Experiments** |  |
| 1 | Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP |  |
| 2 | Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism. |  |
| 3 | Take an example subnet of hosts and obtain a broadcast tree for the subnet. |  |
| 4 | Implement distance vector routing algorithm for obtaining routing tables at each node. |  |
| 5 | Design the following   1. TCP iterative Client and server application to reverse the given input sentence. 2. TCP client and server application to transfer file. 3. TCP concurrent server to convert a given text into upper case using multiplexing system call “select”. 4. TCP concurrent server to echo given set of sentences using poll functions. |  |
| 6 | Design the following  a) UDP Client and server application to reverse the given input sentence. 2018-2019 173.  b) UDP Client server to transfer a file. |  |
| 7 | Programs to demonstrate the usage of Advanced socket system calls like getsockopt( ), setsockopt(), getpeername ( ), getsockname( ), readv( ) and  writev( ). |  |
| 8 | Implementation of concurrent chat server that allows current logged in users to communicate one with other. |  |
| 9 | Implementation of DNS. |  |
| 10 | Implementation of Ping service. |  |

**KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY**

**(AN AUTONOMOUS INSTITUTION)**

### Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH,

### Narayanguda, Hyderabad – 500029

**Department of Computer Science & Engineering**

**Vision of the Institution:**

To be the fountainhead of latest technologies, producing highly skilled, globally competent engineers.

### Mission of the Institution:

* To provide a learning environment that inculcates problem solving skills, professional, ethical responsibilities, lifelong learning through multi modal platforms and prepare students to become successful professionals.
* To establish Industry Institute Interaction to make students ready for the industry.
* To provide exposure to students on latest hardware and software tools.
* To promote research-based projects/activities in the emerging areas of technology convergence.
* To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises
* To induce a spirit of nationalism which will enable the student to develop, understand India’s challenges and to encourage them to develop effective solutions.
* To support the faculty to accelerate their learning curve to deliver excellent service to students.

# KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

## (AN AUTONOMOUS INSTITUTE)

### Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Narayanguda, Hyderabad – 500029

**Department of Computer Science & Engineering**

**Vision of the Department**:

To be among the region's premier teaching and research Computer Science and Engineering departments producing globally competent and socially responsible graduates in the most conducive academic environment.

### Mission of the Department:

* + To provide faculty with state-of-the-art facilities for continuous professional development and research, both in foundational aspects and of relevance to emerging computing trends.
  + To impart skills that transform students to develop technical solutions for societal needs and inculcate entrepreneurial talents.
  + To inculcate an ability in students to pursue the advancement of knowledge in various specializations of Computer Science and Engineering and make them industry-ready.
  + To engage in collaborative research with academia and industry and generate adequate resources for research activities for seamless transfer of knowledge resulting in sponsored projects and consultancy.
  + To cultivate responsibility through sharing of knowledge and innovative computing solutions that benefits the society-at-large.
  + To collaborate with academia, industry and community to set high standards in academic excellence and in fulfilling societal responsibilities.

# KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

## (AN AUTONOMOUS INSTITUTE)

### Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Narayanguda, Hyderabad – 500029

**Department of Computer Science & Engineering**

**PROGRAM OUTCOMES (POs)**

**PO1: Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2: Problem Analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO3: 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.

**PO4: 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.

**PO5: 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.

**PO6: 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.

**PO7: 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.

**PO8: Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9: Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO10: 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.

**PO11: 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.

**PO12: 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.

# KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

## (AN AUTONOMOUS INSTITUTE)

### Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Narayanguda, Hyderabad – 500029

**Department of Computer Science & Engineering**

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO1**: An ability to analyze the common business functions to design and develop appropriate Computer Science solutions for social upliftment.

**PSO2**: Shall have expertise on the evolving technologies like Python, Machine Learning, Deep Learning, Internet of Things (IOT), Data Science, Full stack development, Social Networks, Cyber Security, Big Data, Mobile Apps, CRM, ERP etc.

# KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

## (AN AUTONOMOUS INSTITUTE)

### Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Narayanguda, Hyderabad – 500029

**Department of Computer Science & Engineering**

**PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO1:** Graduates will have successful careers in computer related engineering fields or will be able to successfully pursue advanced higher education degrees.

**PEO2:** Graduates will try and provide solutions to challenging problems in their profession by applying computer engineering principles.

**PEO3:** Graduates will engage in life-long learning and professional development by rapidly adapting changing work environment.

**PEO4:** Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism and ethical responsibility.

# C:\KMIT EXAMINATION BRANCH-AUTONOMOUS\DESK TOP DOCUMENTS\kmit-bar.pngKESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTE)

Accredited by NBA& NAAC, Approved by AICTE, Affiliated to JNTUH, Hyderabad

## B. Tech. in COMPUTER SCIENCE AND ENGINEERING

## III Year I Semester Syllabus (KR21)

**COMPUTER NETWORKS LAB (21CS502PC)**

## Common to CSE, IT, CSE(AI&ML) and CSE(DS)

|  |  |  |  |
| --- | --- | --- | --- |
| **L** | **T** | **P** | **C** |
| **0** | **0** | **3** | **1.5** |

**Pre-requisite/ Co-requisites:**

* 1. PP102ES - Programming for problem solving Course.
  2. 21CS401PC - Java Programming Course.
  3. 21CS501PC – Computer Networks Course.

**Course Objectives: The course will help to**

1. Introduce CRC Mechanism.
2. Understand the concepts of Data link layer.
3. Gain the knowledge on network layer.
4. Understand the concepts of TCP and UDP Protocols.
5. Learn the concepts of sockets and DNS.

**Course Outcomes: After learning the concepts of this course, the student is able to**

1. Compute CRC Mechanisms.
2. Demonstrate and implement the Go-Back-N mechanism.
3. Demonstrate and Apply routing algorithms.
4. Illustrate and implement TCP and UDP Client and server Applications.
5. Develop DNS and Ping service.

**Software to be used: The students can use any OS with Java.**

**List of Programs: Using C/Java programming**

1. Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP
2. Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.
3. Take an example subnet of hosts and obtain a broadcast tree for the subnet.
4. Implement distance vector routing algorithm for obtaining routing tables at each node.
5. Design the following
   1. TCP iterative Client and server application to reverse the given input sentence.
   2. TCP client and server application to transfer file.
   3. TCP concurrent server to convert a given text into upper case using multiplexing system call “select”.
   4. TCP concurrent server to echo given set of sentences using poll functions.
6. Design the following
   1. UDP Client and server application to reverse the given input sentence. 2018-2019 173.
   2. UDP Client server to transfer a file.
7. Programs to demonstrate the usage of Advanced socket system calls like getsockopt( ), setsockopt(), getpeername ( ),getsockname( ),readv( ) and writev( ).
8. Implementation of concurrent chat server that allows current logged in users to communicate one with other.
9. Implementation of DNS.
10. Implementation of Ping service.

**TEXT BOOKS:**

1. Data Communications and Networking-Behrouz A. Forouzan, Fourth Edition TMH,2006.
2. Computer Networks- Andrew S Tanenbaum, 4th Edition. Pearson Education,PHI.
3. UNIX Network Programming-W.Richard Stevens, Bill Fenner, Andrew M. Rudoff, Pearson Education.
4. UNIX Network Programming- – W. Richard Stevens, PHI 1st Edition.

**REFERENCE BOOKS:**

1. Data communications and Computer Networks- P.C Gupta, PHI.
2. An Engineering Approach to Computer Networks-S. Keshav, 2nd Edition, Pearson Education.
3. Understanding communications and Networks- W.A. Shay, C engage Learning 3rd Edition.
4. Computer Networking: A Top-Down Approach Featuring the Internet-James F.Kurose & Keith W. Ross, 3rdEdition, Pearson Education.
5. Data and Computer Communication-William Stallings, Pearson Education,6thEdition ,2000.
6. UNIX for Programmers and Users- Graham GLASS, King abls, Pearson Education 3rd Edition.
7. Advanced UNIX Programming- M. J. ROCHKIND, Pearson Education, 2nd Edition.

****KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTE)

Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Hyderabad

## DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

**Course Objectives:**

* Introduce CRC Mechanism.
* Understand the concepts of Data link layer.
* Gain the knowledge on network layer.
* Understand the concepts of TCP and UDP Protocols.
* Learn the concepts of sockets and DNS.

**Course Outcomes:**

After learning the contents of this course, the student is able to

CO 1: Compute CRC Mechanisms.

CO 2: Demonstrate and implement the Go-Back-N mechanism.

CO 3: Demonstrate and Apply routing algorithms.

CO 4: Illustrate and implement TCP and UDP Client and server Applications.

CO 5: Develop DNS and Ping service.

**CO-PO MAPPING:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CO** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** |
| **Computer Networks Lab** | **CO1** | 1 | 2 |  | 3 |  |  |  |  |  |  |  |  |
| **CO2** |  | 1 |  | 2 |  |  |  |  |  |  |  |  |
| **CO3** |  | 2 | 1 | 2 | 3 |  |  |  |  |  |  |  |
| **CO4** |  | 1 |  |  | 2 |  |  |  |  |  |  |  |
| **CO5** | 1 | 1 |  | 2 |  |  |  |  |  |  |  |  |

**CO-PSO MAPPING:**

|  |  |  |
| --- | --- | --- |
|  | **PSO-1** | **PSO-2** |
| **CO1** | 1 |  |
| **CO2** |  |  |
| **CO3** |  | 2 |
| **CO4** |  |  |
| **CO5** | 2 |  |

**EXPERIMENT 1**

**Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP**

**Program:**

import java.lang.\*;

import java.util.\*;

// create CRCExample class to demonstrate the working of Cyclic Redundancy Check

class CRCExample {

// main() method start

public static void main(String args[]) {

// create scanner class object to take input from user

Scanner scan = new Scanner(System.in);

// declare n for the size of the data

int size;

// take the size of the data from the user

System.out.println("Enter the size of the data array: ");

size = scan.nextInt();

// declaration of the data array

int data[] = new int[size];

// take bits of the data from the user

System.out.println("Enter data bits in the array one by one: ");

for(int i = 0 ; i < size ; i++) {

System.out.println("Enter bit " + (size-i) + ":");

data[i] = scan.nextInt();

}

// take the size of the divisor from the user

System.out.println("Enter the size of the divisor array:");

size = scan.nextInt();

// declaration of the divisor array

int divisor[] = new int[size];

System.out.println("Enter divisor bits in the array one by one: ");

for(int i = 0 ; i < size ; i++) {

System.out.println("Enter bit " + (size-i) + ":");

divisor[i] = scan.nextInt();

}

// Divide the input data by the input divisor and store the result in the rem array

int rem[] = divideDataWithDivisor(data, divisor);

// iterate rem using for loop to print each bit

for(int i = 0; i < rem.length-1; i++) {

System.out.print(rem[i]);

}

System.out.println("\nGenerated CRC code is: ");

for(int i = 0; i < data.length; i++) {

System.out.print(data[i]);

}

for(int i = 0; i < rem.length-1; i++) {

System.out.print(rem[i]);

}

System.out.println();

// we create a new array that contains the original data with its CRC code

// the size of the sentData array with be equal to the sum of the data and the rem arrays length

int sentData[] = new int[data.length + rem.length - 1];

System.out.println("Enter bits in the array which you want to send: ");

for(int i = 0; i < sentData.length; i++) {

System.out.println("Enter bit " +(sentData.length - 1)+ ":");

sentData[i] = scan.nextInt();

}

receiveData(sentData, divisor);

}

// create divideDataWithDivisor() method to get CRC

static int[] divideDataWithDivisor(int oldData[], int divisor[]) {

// declare rem[] array

int rem[] = new int[divisor.length];

int i;

int data[] = new int[oldData.length + divisor.length];

// use system's arraycopy() method for copying data into rem and data arrays

System.arraycopy(oldData, 0, data, 0, oldData.length);

System.arraycopy(data, 0, rem, 0, divisor.length);

// iterate the oldData and exor the bits of the remainder and the divisor

for(i = 0; i < oldData.length; i++) {

System.out.println((i+1) + ".) First data bit is : "+ rem[0]);

System.out.print("Remainder : ");

if(rem[0] == 1) {

// We have to exor the remainder bits with divisor bits

for(int j = 1; j < divisor.length; j++) {

rem[j-1] = exorOperation(rem[j], divisor[j]);

System.out.print(rem[j-1]);

}

}

else {

// We have to exor the remainder bits with 0

for(int j = 1; j < divisor.length; j++) {

rem[j-1] = exorOperation(rem[j], 0);

System.out.print(rem[j-1]);

}

}

// The last bit of the remainder will be taken from the data

// This is the 'carry' taken from the dividend after every step

// of division

rem[divisor.length-1] = data[i+divisor.length];

System.out.println(rem[divisor.length-1]);

}

return rem;

}

// create exorOperation() method to perform exor data

static int exorOperation(int x, int y) {

// This simple function returns the exor of two bits

if(x == y) {

return 0;

}

return 1;

}

// method to print received data

static void receiveData(int data[], int divisor[]) {

int rem[] = divideDataWithDivisor(data, divisor);

// Division is done

for(int i = 0; i < rem.length; i++) {

if(rem[i] != 0) {

// if the remainder is not equal to zero, data is currupted

System.out.println("Currupted data received...");

return;

}

}

System.out.println("Data received without any error.");

}

}

**OUTPUT:**

**EXPERIMENT 2**

1. **Develop a simple data link layer that performs the flow control using the sliding window protocol.**

**Program:**

import java.util.Scanner;

public class SlidingWindowProtocol {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n, f, i;

System.out.print("Enter window size: ");

n = scanner.nextInt();

System.out.print("Enter number of frames to transmit: ");

f = scanner.nextInt();

int[] frames = new int[f + 1]; // Adding 1 to the array size for 1-based indexing

System.out.println("Enter " + f + " frames:");

for (i = 1; i <= f; i++) {

frames[i] = scanner.nextInt();

}

System.out.println("\nWith sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)\n");

System.out.println("After sending " + n + " frames at each stage sender waits for acknowledgement sent by the receiver\n");

for (i = 1; i <= f; i++) {

if (i % n == 0) {

System.out.print(frames[i] + " ");

System.out.println("\n Acknowledgment of above frames sent is received by sender\n");

} else {

System.out.print(frames[i] + " ");

}

}

if (f % n != 0) {

System.out.println("\n Acknowledgment of above frames sent is received by sender");

}

}

}

**OUTPUT:**

**b) Develop a simple data link layer that performs the loss recovery using the Go-Back-N mechanism.**

**Program:**

import java.lang.\*;

import java.io.\*;

public class GoBackN {

public static void main(String args[]) throws IOException

{

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

System.out.println("Please enter the Window Size: ");

int window = Integer.parseInt(br.readLine());

boolean loop = true;

int sent = 0;

while(loop)

{

for(int i = 0; i < window; i++)

{

System.out.println("Frame " + sent + " has been transmitted.");

sent++;

if(sent == window)

break;

}

System.out.println("Please enter the last Acknowledgement received.");

int ack = Integer.parseInt(br.readLine());

if(ack == window)

loop = false;

else

sent = ack;

}

}

}

**OUTPUT:**

**EXPERIMENT-3**

**Take an example subnet of hosts and obtain a broadcast tree for the subnet.**

**Program:**

import java.util.\*;

import java.io.\*;

class SubnettingClassful {

public static void main(String args[]) {

System.out.println("Enter the Network address");

Scanner sc = new Scanner(System.in);

String ip = sc.next();

String[] IPs = ip.split("\\.");

int[] IP = new int[IPs.length];

for (int i = 0; i<IPs.length; i++) {

IP[i] = Integer.parseInt(IPs[i]);

//System.out.println(IP[i]);

}

System.out.println("Enter the number of Subnets");

int sub = sc.nextInt();

int temp = IP[0];

int mask;

if (temp < 128) {

mask = 8;

} else if (temp < 192) {

mask = 16;

} else if (temp < 224) {

mask = 24;

} else {

System.out.println("Invalid IP for subnetting");

return;

}

int rem = 32 - mask;

long incr1 = 0;

long incr = ((long) Math.round(Math.pow(2, rem))) / sub;

long incr2 = incr;

long t1 = 0;

//System.out.println(rem+" "+incr1+"\t"+incr2);

for (int j = 0; j < sub; j++) {

t1 = incr2;

//System.out.println(incr2);

System.out.print(IP[0]);

System.out.print("." + (IP[1] + (incr1 / (256 \* 256))) % 256);

System.out.print("." + (IP[2] + (incr1 / 256)) % 256);

System.out.print("." + (IP[3] + (incr1 % 256)));

System.out.print("\t-TO-\t");

System.out.print(IP[0]);

System.out.print("." + (IP[1] + (t1 / (256 \* 256)) - 1) % 256);

System.out.print("." + (IP[2] + (t1 / 256) - 1) % 256);

System.out.print("." + (IP[3] + ((t1 - 1) % 256)));

System.out.println("\n");

incr1 = incr2;

incr2 = incr1 + incr;

}

}

}

**OUTPUT:**

**EXPERIMENT-4**

**Implement distance vector routing algorithm for obtaining routing tables at each node.**

**Program:**

import java.io.\*;

public class DVR

{

static int graph[][];

static int via[][];

static int rt[][];

static int v;

static int e;

public static void main(String args[]) throws IOException

{

BufferedReaderbr = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Please enter the number of Vertices: ");

v = Integer.parseInt(br.readLine());

System.out.println("Please enter the number of Edges: ");

e = Integer.parseInt(br.readLine());

graph = new int[v][v];

via = new int[v][v];

rt = new int[v][v];

for(int i = 0; i< v; i++)

for(int j = 0; j < v; j++)

{

if(i == j)

graph[i][j] = 0;

else

graph[i][j] = 9999;

}

for(int i = 0; i< e; i++)

{

System.out.println("Please enter data for Edge " + (i + 1) + ":");

System.out.print("Source: ");

int s = Integer.parseInt(br.readLine());

s--;

System.out.print("Destination: ");

int d = Integer.parseInt(br.readLine());

d--;

System.out.print("Cost: ");

int c = Integer.parseInt(br.readLine());

graph[s][d] = c;

graph[d][s] = c;

}

dvr\_calc\_disp("The initial Routing Tables are: ");

System.out.print("Please enter the Source Node for the edge whose cost has changed: ");

int s = Integer.parseInt(br.readLine());

s--;

System.out.print("Please enter the Destination Node for the edge whose cost has changed: ");

int d = Integer.parseInt(br.readLine());

d--;

System.out.print("Please enter the new cost: ");

int c = Integer.parseInt(br.readLine());

graph[s][d] = c;

graph[d][s] = c;

dvr\_calc\_disp("The new Routing Tables are: ");

}

static void dvr\_calc\_disp(String message)

{

System.out.println();

init\_tables();

update\_tables();

System.out.println(message);

print\_tables();

System.out.println();

}

static void update\_table(int source)

{

for(int i = 0; i< v; i++)

{

if(graph[source][i] != 9999)

{

int dist = graph[source][i];

for(int j = 0; j < v; j++)

{

int inter\_dist = rt[i][j];

if(via[i][j] == source)

inter\_dist = 9999;

if(dist + inter\_dist< rt[source][j])

{

rt[source][j] = dist + inter\_dist;

via[source][j] = i;

}

}

}

}

}

static void update\_tables()

{

int k = 0;

for(int i = 0; i< 4\*v; i++)

{

update\_table(k);

k++;

if(k == v)

k = 0;

}

}

static void init\_tables()

{

for(int i = 0; i< v; i++)

{

for(int j = 0; j < v; j++)

{

if(i == j)

{

rt[i][j] = 0;

via[i][j] = i;

}

else

{

rt[i][j] = 9999;

via[i][j] = 100;

}

}

}

}

static void print\_tables()

{

for(int i = 0; i< v; i++)

{

for(int j = 0; j < v; j++)

{

System.out.print("Dist: " + rt[i][j] + " ");

}

System.out.println();

}

}

}

**OUTPUT:**

**EXPERIMENT-5**

**Design the following  
a. TCP iterative Client and server application to reverse the given input sentence.**

**Server Program:**

import java.net.\*;

import java.io.\*;

public class ServerRevString{

public static void main(String arg[]) throws Exception{

ServerSocket server = new ServerSocket(1234);

System.out.println("Server is Waiting");

while(true){

Socket con = server.accept();

DataInputStream in = new DataInputStream(con.getInputStream());

DataOutputStream out = new DataOutputStream(con.getOutputStream());

StringBuilder inp = new StringBuilder(in.readUTF().toString());

StringBuilder op=inp.reverse();

out.writeUTF(op.toString());

System.out.println(op.toString());

}

}

}

**Client Program:**

import java.net.\*;

import java.io.\*;

import java.util.Scanner;

public class ClientRevString{

public static void main(String arg[]) throws Exception{

InetAddressia = InetAddress.getLocalHost();

Socket cSock = new Socket(ia,1234);

DataInputStream in = new DataInputStream(cSock.getInputStream());

DataOutputStream out = new DataOutputStream(cSock.getOutputStream());

System.out.println("Please Enter String");

Scanner sc = new Scanner(System.in);

String inp = sc.nextLine();

out.writeUTF(inp);

// System.out.println("response from server");

// System.out.println(in.readUTF().toString());

cSock.close();

}

}

**OUTPUT:**

**b. TCP client and server application to transfer file.**

**Program**

**File Server :**

import java.io.BufferedInputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.OutputStream;

import java.net.InetAddress;

import java.net.ServerSocket;

import java.net.Socket;

public class FileServer

{

public static void main(String[] args) throws Exception

{

//Initialize Sockets

ServerSocketssock = new ServerSocket(5000);

Socket socket = ssock.accept();

//The InetAddress specification

InetAddress IA = InetAddress.getByName("localhost");

//Specify the file

File file = new File("e:\\Bookmarks.html");

FileInputStreamfis = new FileInputStream(file);

BufferedInputStream bis = new BufferedInputStream(fis);

//Get socket's output stream

OutputStreamos = socket.getOutputStream();

//Read File Contents into contents array

byte[] contents;

long fileLength = file.length();

long current = 0;

long start = System.nanoTime();

while(current!=fileLength){

int size = 10000;

if(fileLength - current >= size)

current += size;

else{

size = (int)(fileLength - current);

current = fileLength;

}

contents = new byte[size];

bis.read(contents, 0, size);

os.write(contents);

System.out.print("Sending file ... "+(current\*100)/fileLength+"% complete!");

}

os.flush();

//File transfer done. Close the socket connection!

socket.close();

ssock.close();

System.out.println("File sent succesfully!");

} }

**File Client:**

import java.io.BufferedOutputStream;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.net.InetAddress;

import java.net.Socket;

public class FileClient {

public static void main(String[] args) throws Exception{

//Initialize socket

Socket socket = new Socket(InetAddress.getByName("localhost"), 5000);

byte[] contents = new byte[10000];

//Initialize the FileOutputStream to the output file's full path.

FileOutputStreamfos = new FileOutputStream("e:\\Bookmarks1.html");

BufferedOutputStreambos = new BufferedOutputStream(fos);

InputStream is = socket.getInputStream();

//No of bytes read in one read() call

int bytesRead = 0;

while((bytesRead=is.read(contents))!=-1)

bos.write(contents, 0, bytesRead);

bos.flush();

socket.close();

System.out.println("File saved successfully!");

}

}

**OUTPUT:**

**c. TCP concurrent server to convert a given text into upper case using multiplexing system call “select”.**

**Program:**

**TCP Server:**import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.nio.ByteBuffer;

import java.nio.channels.SelectionKey;

import java.nio.channels.Selector;

import java.nio.channels.ServerSocketChannel;

import java.nio.channels.SocketChannel;

import java.util.Iterator;

import java.util.Set;

public class TCPServer {

private static final int PORT = 5555;

public static void main(String[] args) {

try {

Selector selector = Selector.open();

ServerSocketChannelserverSocketChannel = ServerSocketChannel.open();

ServerSocketserverSocket = serverSocketChannel.socket();

serverSocket.bind(new java.net.InetSocketAddress(PORT));

serverSocketChannel.configureBlocking(false);

serverSocketChannel.register(selector, SelectionKey.OP\_ACCEPT);

System.out.println("Server is listening on port " + PORT);

while (true) {

int readyChannels = selector.select();

if (readyChannels == 0) continue;

Set<SelectionKey>selectedKeys = selector.selectedKeys();

Iterator<SelectionKey>keyIterator = selectedKeys.iterator();

while (keyIterator.hasNext()) {

SelectionKey key = keyIterator.next();

if (key.isAcceptable()) {

acceptConnection(selector, serverSocketChannel);

} else if (key.isReadable()) {

processRequest(key);

}

keyIterator.remove();

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

private static void acceptConnection(Selector selector, ServerSocketChannelserverSocketChannel) throws IOException {

SocketChannelclientChannel = serverSocketChannel.accept();

clientChannel.configureBlocking(false);

clientChannel.register(selector, SelectionKey.OP\_READ);

System.out.println("Accepted connection from " + clientChannel.getRemoteAddress());

}

private static void processRequest(SelectionKey key) throws IOException {

SocketChannelclientChannel = (SocketChannel) key.channel();

ByteBuffer buffer = ByteBuffer.allocate(1024);

int bytesRead = clientChannel.read(buffer);

if (bytesRead == -1) {

key.cancel();

clientChannel.close();

return;

}

String request = new String(buffer.array(), 0, bytesRead);

System.out.println("Received request from " + clientChannel.getRemoteAddress() + ": " + request);

String response = request.toUpperCase();

clientChannel.write(ByteBuffer.wrap(response.getBytes()));

key.cancel();

clientChannel.close();

System.out.println("Connection closed by client: " + clientChannel.getRemoteAddress());

}

}

**TCP CLIENT:**

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.SocketChannel;

import java.util.Scanner;

public class TCPClient {

private static final String SERVER\_IP = "localhost";

private static final int SERVER\_PORT = 5555;

public static void main(String[] args) {

try {

InetSocketAddressserverAddress = new InetSocketAddress(SERVER\_IP, SERVER\_PORT);

SocketChannelsocketChannel = SocketChannel.open(serverAddress);

Scanner scanner = new Scanner(System.in);

System.out.print("Enter text to convert to uppercase: ");

String userInput = scanner.nextLine();

socketChannel.write(ByteBuffer.wrap(userInput.getBytes()));

ByteBuffer buffer = ByteBuffer.allocate(1024);

socketChannel.read(buffer);

String response = new String(buffer.array()).trim();

System.out.println("Server response: " + response);

socketChannel.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

**OUTPUT:**

**d. Write a program using TCP Concurrenct Server to echo given set of sentences using poll function.**

**Program:**

**Server:**

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class TCPServer {

private static final int PORT = 5555;

public static void main(String[] args) {

try {

ServerSocketserverSocket = new ServerSocket(PORT);

System.out.println("Server is listening on port " + PORT);

ExecutorServiceexecutorService = Executors.newFixedThreadPool(10);

while (true) {

Socket clientSocket = serverSocket.accept();

System.out.println("Accepted connection from " + clientSocket.getRemoteSocketAddress());

executorService.execute(new ClientHandler(clientSocket));

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

class ClientHandler implements Runnable {

private Socket clientSocket;

public ClientHandler(Socket clientSocket) {

this.clientSocket = clientSocket;

}

@Override

public void run() {

try (

BufferedReader reader = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true)

) {

String line;

while ((line = reader.readLine()) != null) {

System.out.println("Received from " + clientSocket.getRemoteSocketAddress() + ": " + line);

writer.println("Server Echo: " + line);

}

System.out.println("Connection closed by client: " + clientSocket.getRemoteSocketAddress());

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

clientSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

**Client :**

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.Socket;

import java.util.Arrays;

import java.util.List;

public class TCPClient {

private static final String SERVER\_IP = "localhost";

private static final int SERVER\_PORT = 5555;

public static void main(String[] args) {

List<String> sentences = Arrays.asList(

"Hello, Server!",

"How are you?",

"Echo this sentence.",

"TCP is awesome!"

);

try (Socket socket = new Socket(SERVER\_IP, SERVER\_PORT);

PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);

BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()))

) {

for (String sentence : sentences) {

writer.println(sentence);

String response = reader.readLine();

System.out.println("Server response: " + response);

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

**OUTPUT:**

**EXPERIMENT-6**

**Design the following  
a. UDP Client and server application to reverse the given input sentence 2018-2019 173.**

**Program:**

**Client:**import java.io.\*;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class ClientReverse {

public static void main(String[] args) throws Exception{

BufferedReaderbr = new BufferedReader(new InputStreamReader(System.in));

DatagramSocketclientSocket = new DatagramSocket();

InetAddressIPAdress = InetAddress.getLoopbackAddress();

byte [] sendData = new byte[1024];

byte [] receiveData = new byte[1024];

String str = br.readLine();

sendData = str.getBytes();

DatagramPacketdgp = new DatagramPacket(sendData,sendData.length,IPAdress,9999);

clientSocket.send(dgp);

dgp = new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(dgp);

str = new String(dgp.getData());

System.out.println(\"Output: \" + str);

clientSocket.close();

br.close();

}

}

**Server.java:**

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class ServerReverse {

public static String reverseInt(int num) {

int ans = 0;

while (num > 0) {

ans = ans \* 10 + (num % 10);

num = num / 10;

}

return String.valueOf(ans);

}

public static void main(String[] args) throws Exception {

DatagramSocketdgs = new DatagramSocket(9999);

byte[] receiveData = new byte[1024];

byte[] sendData = new byte[1024];

DatagramPacketdgp;

dgp = new DatagramPacket(receiveData, receiveData.length);

dgs.receive(dgp);

String str = new String(dgp.getData());

System.out.println(\"Data Received: \" + str);

InetAddressIPAddress = dgp.getAddress();

String ans = ServerReverse.reverseInt(Integer.parseInt(str.toString()));

sendData = ans.getBytes();

dgp = new DatagramPacket(sendData, sendData.length, IPAddress, dgp.getPort());

dgs.send(dgp);

}

}

**OUTPUT:**

**b. UDP Client server to transfer a file.**

**Program:**

**/\*CLIENT \*/**

import java.net.\*;

import java.io.\*;

public class client

{

public static void main(String args[])throws Exception

{

byte b[]=new byte[1024];

FileInputStream f=new FileInputStream("D:/raj.txt");

DatagramSocket dsoc=new DatagramSocket(2000);

int i=0;

while(f.available()!=0)

{

b[i]=(byte)f.read();

i++;

}

f.close();

dsoc.send(new DatagramPacket(b,i,InetAddress.getLocalHost(),1000));

}

}

**/\*SERVER \*/**

import java.net.\*;

import java.io.\*;

public class server

{

public static void main(String args[])throws IOException

{

byte b[]=new byte[3072];

DatagramSocket dsoc=new DatagramSocket(1000);

FileOutputStream f=new FileOutputStream("D:/nandha.txt");

while(true)

{

DatagramPacket dp=new DatagramPacket(b,b.length);

dsoc.receive(dp);

System.out.println(new String(dp.getData(),0,dp.getLength()));

}

**OUTPUT**

**EXPERIMENT-7**

**Programs to demonstrate the usage of Advanced socket system calls like getsockopt( ), setsockopt(), getpeername (), getsockname( ), readv( ) and writev( ).**

**Program:**

**Server:**

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.net.ServerSocket;

import java.net.Socket;

public class AdvancedSocketServer {

    public static void main(String[] args) {

        try {

            ServerSocket serverSocket = new ServerSocket(8087);

            System.out.println("Server listening on port 8087...");

            // Accept incoming connections

            Socket clientSocket = serverSocket.accept();

            System.out.println("Connection accepted from: " + clientSocket.getInetAddress());

            // Demonstrate getsockname()

            System.out.println("Server socket local address: " + serverSocket.getLocalSocketAddress());

            // Demonstrate getpeername()

            System.out.println("Client socket remote address: " + clientSocket.getRemoteSocketAddress());

            // Demonstrate setsockopt() and getsockopt()

            clientSocket.setTcpNoDelay(true); // Disabling Nagle's algorithm

            System.out.println("TCP No Delay option: " + clientSocket.getTcpNoDelay());

            // Read data using readv()

            InputStream inputStream = clientSocket.getInputStream();

            byte[] buffer1 = new byte[5];

            byte[] buffer2 = new byte[5];

            int bytesRead = inputStream.read(buffer1);

            bytesRead += inputStream.read(buffer2);

            System.out.println("Data read using readv(): " + new String(buffer1) + new String(buffer2));

            // Write data using writev()

            OutputStream outputStream = clientSocket.getOutputStream();

            byte[] data1 = "Hello".getBytes();

            byte[] data2 = "World".getBytes();

            outputStream.write(data1);

            outputStream.write(data2);

            System.out.println("Data written using writev(): Hello World");

            // Close the sockets

            clientSocket.close();

            serverSocket.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

**Client:**

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.net.Socket;

public class AdvancedSocketClient {

    public static void main(String[] args) {

        try {

            Socket socket = new Socket("localhost", 8080);

            // Demonstrate getsockname()

            System.out.println("Client socket local address: " + socket.getLocalSocketAddress());

            // Demonstrate getpeername()

            System.out.println("Server socket remote address: " + socket.getRemoteSocketAddress());

            // Demonstrate setsockopt() and getsockopt()

            socket.setTcpNoDelay(true); // Disabling Nagle's algorithm

            System.out.println("TCP No Delay option: " + socket.getTcpNoDelay());

            // Write data using writev()

            OutputStream outputStream = socket.getOutputStream();

            byte[] data1 = "Java".getBytes();

            byte[] data2 = "Socket".getBytes();

            outputStream.write(data1);

            outputStream.write(data2);

            System.out.println("Data written using writev(): Java Socket");

            // Read data using readv()

            InputStream inputStream = socket.getInputStream();

            byte[] buffer1 = new byte[4];

            byte[] buffer2 = new byte[6];

            int bytesRead = inputStream.read(buffer1);

            bytesRead += inputStream.read(buffer2);

            System.out.println("Data read using readv(): " + new String(buffer1) + new String(buffer2));

            // Close the socket

            socket.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

**OUTPUT:**

**EXPERIMENT-8**

**Implementation of concurrent chat server that allows current logged in users to communicate one with other.**

**Program:**

**Chat Server**:

import java.io.\*;

import java.net.\*;

public class TCPChatServer {

    private static final int PORT = 8089;

    public static void main(String[] args) {

        try {

            ServerSocket serverSocket = new ServerSocket(PORT);

            System.out.println("Server listening on port " + PORT + "...");

            Socket clientSocket = serverSocket.accept();

            System.out.println("Connection accepted from: " + clientSocket.getInetAddress());

            BufferedReader reader = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

            PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true);

            String message;

            while ((message = reader.readLine()) != null) {

                System.out.println("From client: " + message + "\t To client: ");

                // Simulating server response

                writer.println("Hello, " + message);

                if (message.trim().equalsIgnoreCase("exit")) {

                    System.out.println("Server Exit...");

                    break;

                }

            }

            clientSocket.close();

            serverSocket.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

Chat Client:

import java.io.\*;

import java.net.\*;

public class TCPChatClient {

    private static final int PORT = 8089;

    public static void main(String[] args) {

        try {

            Socket socket = new Socket("localhost", PORT);

            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

            PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);

            BufferedReader serverReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            String userInput;

            while ((userInput = reader.readLine()) != null) {

                writer.println(userInput);

                // Simulating client response

                String serverResponse = serverReader.readLine();

                System.out.println("From Server: " + serverResponse);

                if (userInput.trim().equalsIgnoreCase("exit")) {

                    System.out.println("Client Exit...");

                    break;

                }

            }

            socket.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

**OUTPUT:**

**EXPERIMENT-9**

**Implementation of DNS.**

**Program:**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class DNS

{

public static void main(String[] args)

{

int n;

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

do

{

System.out.println("\n Menu: \n 1. DNS 2. Reverse DNS 3. Exit \n");

System.out.println("\n Enter your choice");

n = Integer.parseInt(System.console().readLine());

if(n==1)

{

try

{

System.out.println("\n Enter Host Name ");

String hname=in.readLine();

InetAddress address;

address = InetAddress.getByName(hname);

System.out.println("Host Name: " + address.getHostName());

System.out.println("IP: " + address.getHostAddress());

}

catch(IOExceptionioe)

{

ioe.printStackTrace();

}

}

if(n==2)

{

try

{

System.out.println("\n Enter IP address");

String ipstr = in.readLine();

InetAddressia = InetAddress.getByName(ipstr);

System.out.println("IP: "+ipstr);

System.out.println("Host Name: " +ia.getHostName());

}

catch(IOExceptionioe)

{

ioe.printStackTrace();

}

}

}while(!(n==3));

}

}

**OUTPUT:**

**EXPERIMENT-10**

**Implementation of Ping service.**

**Program:**

import java.io.\*;

import java.net.\*;

class pingserver

{

public static void main(String args[])

{

try

{

String str;

System.out.print(" Enter the IP Address to be Ping : ");

BufferedReader buf1=new BufferedReader(new

InputStreamReader(System.in));

String ip=buf1.readLine();

Runtime H=Runtime.getRuntime();

Process p=H.exec("ping " + ip);

InputStream in=p.getInputStream();

BufferedReader buf2=new BufferedReader(new

InputStreamReader(in));

while((str=buf2.readLine())!=null)

{

System.out.println(" " + str);

}

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

**OUTPUT:**