# 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 (AI & ML)

**Vision of the College:**

* To be the fountain head in producing highly skilled, globally competent engineers.
* Producing quality graduates trained in the latest software technologies and related tools and striving to make India a world leader in software products and services.

**Mission of the College:**

* 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 lndia'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.

i

# 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 (AI & ML)

**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 (AI & ML)

**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 (AI & ML)

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

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

**PSO2**: Shall have expertise on the evolving technologies like Mobile Apps, CRM, ERP, Big Data, etc.

## 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.

**EXPERIMENT-1**

1. Write a program to compute CRC code for the polynomials

// Include headers

#include<stdio.h>

#include<string.h>

// length of the generator polynomial

#define N strlen(gen\_poly)

// data to be transmitted and received

char data[28];

// CRC value

char check\_value[28];

// generator polynomial

char gen\_poly[10];

// variables

int data\_length,i,j;

// function that performs XOR operation

void XOR(){

// if both bits are the same, the output is 0

// if the bits are different the output is 1

for(j = 1;j < N; j++)

check\_value[j] = (( check\_value[j] == gen\_poly[j])?'0':'1');

}

void crc(){

// initializing check\_value

for(i=0;i<N;i++)

check\_value[i]=data[i];

do{

// check if the first bit is 1 and calls XOR function

if(check\_value[0]=='1')

XOR();

// Move the bits by 1 position for the next computation

for(j=0;j<N-1;j++)

check\_value[j]=check\_value[j+1];

// appending a bit from data

check\_value[j]=data[i++];

}while(i<=data\_length+N-1);

// loop until the data ends

}

// Function to check for errors on the receiver side

void receiver(){

// get the received data

printf("Enter the received data: ");

scanf("%s", data);

printf("\n-----------------------------\n");

printf("Data received: %s", data);

// Cyclic Redundancy Check

crc();

// Check if the remainder is zero to find the error

for(i=0;(i<N-1) && (check\_value[i]!='1');i++);

if(i<N-1)

printf("\nError detected\n\n");

else

printf("\nNo error detected\n\n");

}

int main()

{

// get the data to be transmitted

printf("\nEnter data to be transmitted: ");

scanf("%s",data);

printf("\n Enter the Generating polynomial: ");

// get the generator polynomial

scanf("%s",gen\_poly);

// find the length of data

data\_length=strlen(data);

// appending n-1 zeros to the data

for(i=data\_length;i<data\_length+N-1;i++)

data[i]='0';

printf("\n----------------------------------------");

// print the data with padded zeros

printf("\n Data padded with n-1 zeros : %s",data);

printf("\n----------------------------------------");

// Cyclic Redundancy Check

crc();

// print the computed check value

printf("\nCRC or Check value is : %s",check\_value);

// Append data with check\_value(CRC)

for(i=data\_length;i<data\_length+N-1;i++)

data[i]=check\_value[i-data\_length];

printf("\n----------------------------------------");

// printing the final data to be sent

printf("\n Final data to be sent : %s",data);

printf("\n----------------------------------------\n");

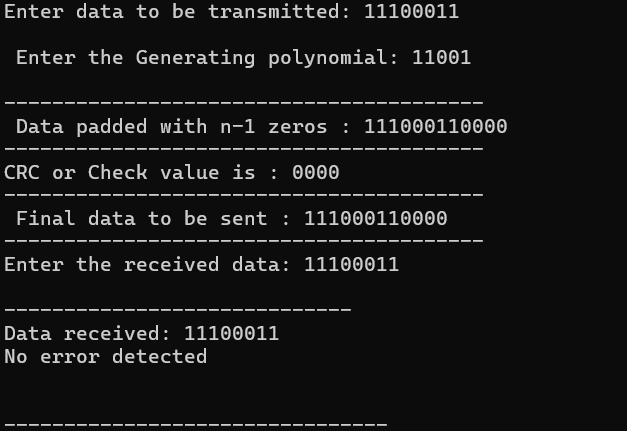
// Calling the receiver function to check errors

receiver();

return 0;

}

OUTPUT:



**EXPERIMENT-2**

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

**AIM:**

**Program:**

#include<stdio.h>

int main(){

int n,f,frames[30],i;

printf("Enter window size : ");

scanf("%d",&n);

printf("Enter number of frames to transmit: ");

scanf("%d",&f);

printf("Enter %d frames: \n",f);

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

scanf("%d",&frames[i]);

}

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

printf("After sending %d frames at each stage sender waits for acknowledgement sent by the receiver\n\n",n);

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

if(i%n==0){

printf("%d ",frames[i]);

printf("\nAcknowledgement of above frames sent is received by sender\n\n");

}

else{

//printf("recieved by sender\n");

printf("%d ",frames[i]);

}

}

if(f%n!=0){

printf("\nAcknowledgement of above frames sent is received by sender\n");

}

}

**Output:**

Enter window size: 3

Enter number of frames to transmit: 5 Enter 5 frames: 12 5 89 4 6

With sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)

After sending 3 frames at each stage sender waits for acknowledgement sent by the receiver 12 5 89

Acknowledgement of above frames sent is received by sender 4 6

Acknowledgement of above frames sent is received by sender

**B. Develop a simple data link layer that performs the flow control using the Go Back N protocol in “c”**

**AIM:**

**Program:**

#include<stdio.h> int main()

{

int window=0;

printf("enter Window size : ");

scanf("%d",&window);

int sent,ack,i=0;

while(1){

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

printf("frame Transmitted %d \n",sent);

sent++;

if(sent==window){

break;

}

}

printf("enter last received acknowledgment : ");

scanf("%d",&ack);

if(ack==window){

break;

}

else{

sent = ack;

}

}

**Output:**

enter window size 8

Frame 0 has been transmitted.

Frame 1 has been transmitted.

Frame 2 has been transmitted.

Frame 3 has been transmitted.

Frame 4 has been transmitted.

Frame 5 has been transmitted.

Frame 6 has been transmitted.

Frame 7 has been transmitted.

Please enter the last Acknowledgement received. 2

Frame 2 has been transmitted.

Frame 3 has been transmitted.

Frame 4 has been transmitted.

Frame 5 has been transmitted.

Frame 6 has been transmitted

Frame 7 has been transmitted

Please enter the last Acknowledgement received. 8

**EXPERIMENT-3**

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

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Function to convert an IP address to a 32-bit integer

unsigned int ipToInt(char\* ip) {

unsigned int a, b, c, d;

sscanf(ip, "%u.%u.%u.%u", &a, &b, &c, &d);

return (a << 24) | (b << 16) | (c << 8) | d;

}

// Function to convert a 32-bit integer to an IP address

void intToIp(unsigned int ip, char\* buffer) {

sprintf(buffer, "%u.%u.%u.%u", (ip >> 24) & 0xFF, (ip >> 16) & 0xFF, (ip >> 8) & 0xFF, ip & 0xFF);

}

// Function to calculate the subnet mask from a prefix length

unsigned int calculateSubnetMask(int prefixLength) {

return prefixLength == 0 ? 0 : ~((1 << (32 - prefixLength)) - 1);

}

int main() {

char ip[16];

int prefixLength, newPrefixLength;

unsigned int subnetMask, newSubnetMask, ipInt;

char buffer[16];

// Input IP address and prefix length

printf("Enter IP address (e.g., 192.168.1.0): ");

scanf("%s", ip);

printf("Enter current prefix length (e.g., 24): ");

scanf("%d", &prefixLength);

// New prefix length for creating two subnets

newPrefixLength = prefixLength + 1;

// Convert IP address to integer

ipInt = ipToInt(ip);

// Calculate original subnet mask and new subnet mask

subnetMask = calculateSubnetMask(prefixLength);

newSubnetMask = calculateSubnetMask(newPrefixLength);

// Calculate the number of hosts per subnet

int hostsPerSubnet = (1 << (32 - newPrefixLength)) - 2; // subtract 2 for network and broadcast addresses

printf("\nNumber of subnets: 2\n");

printf("Number of hosts per subnet: %d\n", hostsPerSubnet);

// Generate subnets

for (int i = 0; i < 2; i++) {

unsigned int subnetNetwork = (ipInt & subnetMask) | (i << (32 - newPrefixLength));

unsigned int subnetBroadcast = subnetNetwork | ~newSubnetMask;

unsigned int firstHost = subnetNetwork + 1;

unsigned int lastHost = subnetBroadcast - 1;

printf("\nSubnet %d:\n", i + 1);

printf("Network Address: ");

intToIp(subnetNetwork, buffer);

printf("%s\n", buffer);

printf("Broadcast Address: ");

intToIp(subnetBroadcast, buffer);

printf("%s\n", buffer);

printf("Subnet Mask: ");

intToIp(newSubnetMask, buffer);

printf("%s\n", buffer);

printf("First Host: ");

intToIp(firstHost, buffer);

printf("%s\n", buffer);

printf("Last Host: ");

intToIp(lastHost, buffer);

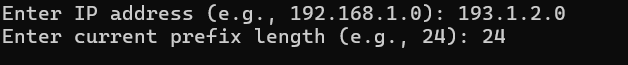
printf("%s\n", buffer);

}

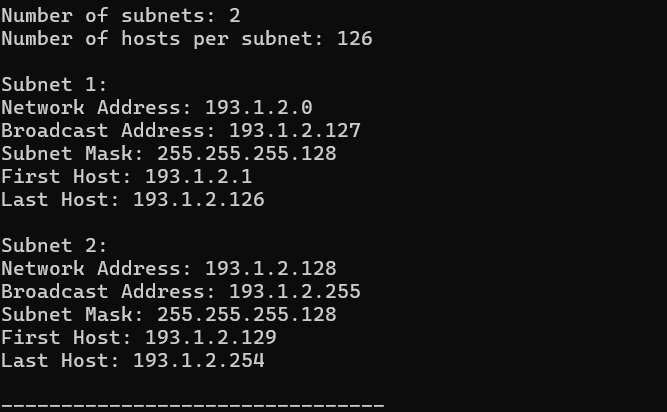
return 0;

}

INPUT:



OUTPUT:



**EXPERIMENT-4**

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

#include <stdio.h>

#include <stdlib.h>

#define INF 9999

#define MAX\_NODES 10

// Function to initialize distance vector and routing table

void initialize(int numNodes, int costMatrix[MAX\_NODES][MAX\_NODES], int distVector[MAX\_NODES][MAX\_NODES], int nextHop[MAX\_NODES][MAX\_NODES]) {

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

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

distVector[i][j] = costMatrix[i][j];

if (costMatrix[i][j] != INF && i != j) {

nextHop[i][j] = j;

} else {

nextHop[i][j] = -1;

}

}

}

}

// Function to print routing table for each node

void printRoutingTable(int numNodes, int distVector[MAX\_NODES][MAX\_NODES], int nextHop[MAX\_NODES][MAX\_NODES]) {

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

printf("Routing table for node %d:\n", i);

printf("Destination\tNext Hop\tDistance\n");

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

if (distVector[i][j] == INF) {

printf("%d\t\t-\t\tINF\n", j);

} else {

printf("%d\t\t%d\t\t%d\n", j, nextHop[i][j], distVector[i][j]);

}

}

printf("\n");

}

}

// Function to implement Distance Vector Routing algorithm

void distanceVectorRouting(int numNodes, int costMatrix[MAX\_NODES][MAX\_NODES], int distVector[MAX\_NODES][MAX\_NODES], int nextHop[MAX\_NODES][MAX\_NODES]) {

int updated;

do {

updated = 0;

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

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

for (int k = 0; k < numNodes; k++) {

if (distVector[i][k] + distVector[k][j] < distVector[i][j]) {

distVector[i][j] = distVector[i][k] + distVector[k][j];

nextHop[i][j] = nextHop[i][k];

updated = 1;

}

}

}

}

} while (updated);

}

int main() {

int numNodes, costMatrix[MAX\_NODES][MAX\_NODES];

int distVector[MAX\_NODES][MAX\_NODES];

int nextHop[MAX\_NODES][MAX\_NODES];

printf("Enter the number of nodes: ");

scanf("%d", &numNodes);

printf("Enter the cost matrix (use %d for INF):\n", INF);

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

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

scanf("%d", &costMatrix[i][j]);

}

}

initialize(numNodes, costMatrix, distVector, nextHop);

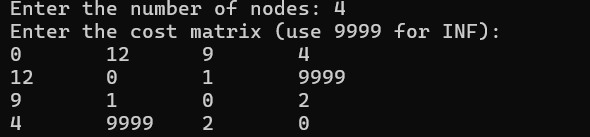
distanceVectorRouting(numNodes, costMatrix, distVector, nextHop);

printRoutingTable(numNodes, distVector, nextHop);

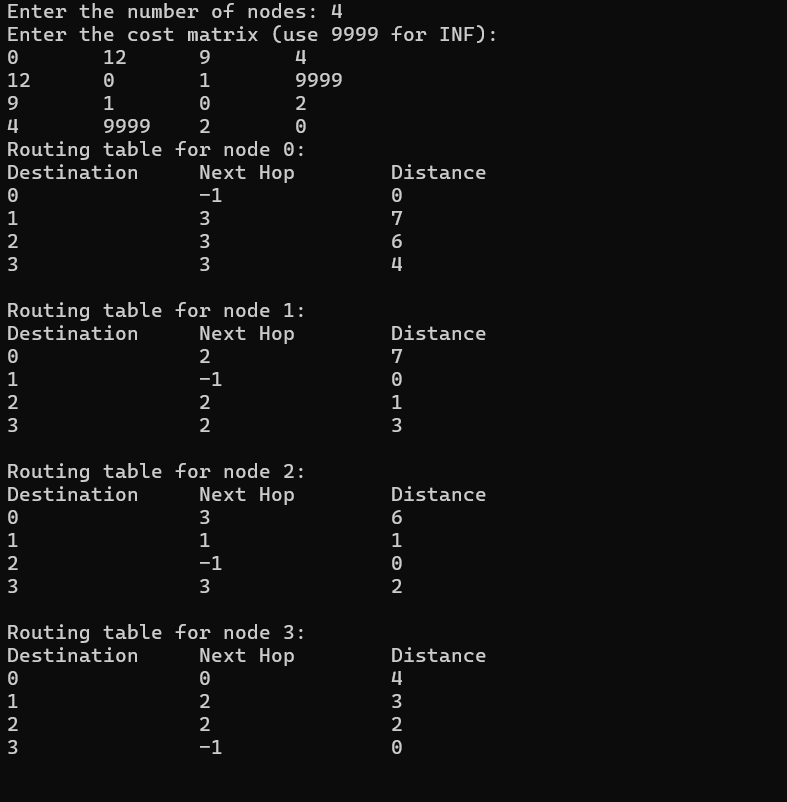
return 0;

}

INPUT:



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:**