**PROGRAM :**

**Server.java**

import java.net.\*;

import java.io.\*;

public class Server

{

private Socket socket = null;

private ServerSocket server = null;

private DataInputStream in = null;

public Server(int port)

{

try

{

server = new ServerSocket(port);

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

System.out.println("Waiting for a client ...");

socket = server.accept();

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

in = new DataInputStream(new BufferedInputStream(socket.getInputStream()));

String line = "";

while (!line.equals("Over"))

{

try

{

line = in.readUTF();

System.out.println(line);

}

catch(IOException i)

{

System.out.println(i);

}

}

System.out.println("Closing connection");

socket.close();

in.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Server server = new Server(5000);

}

}

**Client.java**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class Client

{

private Socket socket = null;

private DataInputStream input = null;

private DataOutputStream out = null;

public Client(String address, int port)

{

try

{

socket = new Socket(address, port);

System.out.println("Connected");

input = new DataInputStream(System.in);

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

}

catch(Exception e)

{

System.out.println(e);

}

String line = "";

Scanner sc = new Scanner(System.in);

while (!line.equals("Over"))

{

try

{

line = sc.next();

out.writeUTF(line);

}

catch(IOException i)

{

System.out.println(i);

}

}

try

{

input.close();

out.close();

socket.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Client client = new Client("127.0.0.1", 5000);

}

}

**SAMPLE INPUT AND OUTPUT :**

|  |  |
| --- | --- |
| Java Server:  Server started  Waiting for a client ...  Client accepted  Hi  I  am  @siddhu\_lol  Over  Closing connection | Java Client:  Connected  Hi  I am @siddhu\_lol  Nice meeting you !!!  Over |

**PROGRAM :**

**WebServer.java**

import java.io.\*;

import java.util.\*;

import java.net.\*;

class WebServer {

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

{

String requestMessageLine;

String fileName;

ServerSocket listenSocket = new ServerSocket(6789);

Socket connectionSocket = listenSocket.accept();

BufferedReader inFromClient = new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));

DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());

requestMessageLine = inFromClient.readLine();

System.out.println(requestMessageLine);

StringTokenizer tokenizedLine = new StringTokenizer(requestMessageLine);

if (tokenizedLine.nextToken().equals("GET"))

{

try

{

fileName = tokenizedLine.nextToken();

if(fileName.startsWith("/") == true)

fileName = fileName.substring(1);

File file = new File(fileName);

int numOfBytes = (int) file.length();

FileInputStream inFile = new FileInputStream(fileName);

byte[] fileInBytes = new byte[numOfBytes];

inFile.read(fileInBytes);

outToClient.writeBytes("HTTP/1.0 200 Document Follows\r\n");

if(fileName.endsWith(".jpg"))

outToClient.writeBytes("Content-Type:image/jpeg\r\n");

if(fileName.endsWith(".gif"))

outToClient.writeBytes("Content-Type:image/gif\r\n");

outToClient.writeBytes("Content-Length: " + numOfBytes + "\r\n");

outToClient.writeBytes("\r\n");

outToClient.write(fileInBytes, 0, numOfBytes);

}

catch(FileNotFoundException e)

{

File file = new File("./404/index.html");

int numOfBytes = (int) file.length();

FileInputStream inFile = new FileInputStream("./404/index.html");

byte[] fileInBytes = new byte[numOfBytes];

inFile.read(fileInBytes);

outToClient.writeBytes("HTTP/1.0 404\r\n");

outToClient.writeBytes("Content-Length: " + numOfBytes + "\r\n");

outToClient.writeBytes("\r\n");

outToClient.write(fileInBytes, 0, numOfBytes);

}

connectionSocket.close();

}

else

System.out.println("Bad Request Message");

}

}

**SAMPLE INPUT AND OUTPUT :**

Request from browser :

http://localhost:6789/image.jpeg

java WebServer:

GET /image.jpeg HTTP/1.1

Header :

**General**

**Request URL:** http://localhost:6789/image.jpeg

**Request Method:** GET

**Status Code:** 200 Document Follows

**Remote Address:** [::1]:6789

**Referrer Policy:** no-referrer-when-downgrade

**Response Headers**

**Content-Length:** 31706

**Request Headers**

**Accept:** text/html,application/xhtml+xml,application/xml; q=0.9,image/webp,image/apng,\*/\*;q=0.8

**Accept-Encoding:** gzip, deflate, br

**Accept-Language:** en-US,en;q=0.9

**Cache-Control:** max-age=0

**Connection:** keep-alive

**Host:** localhost:6789

**Upgrade-Insecure-Requests:** 1

**User-Agent:** Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36

**PROGRAM :**

**Server.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Server

{

public static void main(String args[])

{

try

{

DatagramSocket server=new DatagramSocket(1309);

while(true)

{

byte[] sendbyte=new byte[1024];

byte[] receivebyte=new byte[1024];

DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);

server.receive(receiver);

int n=Integer.parseInt(new String(receiver.getData()).trim());

System.out.println(n);

server.receive(receiver);

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

String s=str.trim();

System.out.println(s);

InetAddress addr=receiver.getAddress();

int port=receiver.getPort();

if(n == 1)

{

InetAddress address;

address = InetAddress.getByName(s);

sendbyte=address.getHostAddress().getBytes();

DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,port);

server.send(sender);

}

if(n == 2)

{

InetAddress ia = InetAddress.getByName(s);

sendbyte=ia.getHostName().getBytes();

DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,port);

server.send(sender);

}

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

**Client.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Client

{

public static void main(String args[])

{

try

{

DatagramSocket client=new DatagramSocket();

InetAddress addr=InetAddress.getByName("127.0.0.1");

byte[] sendbyte=new byte[1024];

byte[] receivebyte=new byte[1024];

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

while(true)

{

System.out.print("Enter your choice : 1. DNS\t 2. Reverse DNS\t 3. Exit\n- -\b\b");

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

if(n==1)

{

sendbyte = Integer.toString(n).getBytes();

DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,1309);

client.send(sender);

System.out.println("Enter the DOMAIN NAME :");

String str=in.readLine();

sendbyte=str.getBytes();

sender=new DatagramPacket(sendbyte,sendbyte.length,addr,1309);

client.send(sender);

DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);

client.receive(receiver);

String s=new String(receiver.getData());

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

}

if(n==2)

{

sendbyte = Integer.toString(n).getBytes();

DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,1309);

client.send(sender);

System.out.println("Enter the IP adress:");

String str=in.readLine();

sendbyte=str.getBytes();

sender=new DatagramPacket(sendbyte,sendbyte.length,addr,1309);

client.send(sender);

DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);

client.receive(receiver);

String s=new String(receiver.getData());

System.out.println("DOMAIN NAME : "+s.trim());

}

if(n == 3)

break;

}

client.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**SAMPLE INPUT AND OUTPUT :**

Java Client:

Enter your choice : 1. DNS 2. Reverse DNS 3. Exit

-1-

Enter the DOMAIN NAME :

www.mitra.mitindia.edu

IP address : 208.113.186.152

Enter your choice : 1. DNS 2. Reverse DNS 3. Exit

-2-

Enter the IP adress:

208.113.186.152

DOMAIN NAME : apache2-lip.grady.dreamhost.com

Enter your choice : 1. DNS 2. Reverse DNS 3. Exit

-3-

Java Server:

[www.mitra.mitindia.edu](http://www.mitra.mitindia.edu)

IP : 208.113.186.152

208.113.186.152

Host : apache2-lip.grady.dreamhost.com

**PROGRAM :**

**Sender.java**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class Sender

{

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

{

ServerSocket s = new ServerSocket(1234);

Socket socket = s.accept();

DataInputStream dis = new DataInputStream(socket.getInputStream());

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

Scanner sc = new Scanner(System.in);

System.out.println("Enter no. of data : ");

int n = sc.nextInt();

dos.writeInt(n);

int checksum = 0;

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

{

int curr = sc.nextInt();

checksum += (curr);

dos.writeInt(curr);

}

System.out.println("Checksum Calculated is : " + ~checksum);

dos.writeInt(~checksum);

}

}

**Reciever.java**

import java.net.\*;

import java.io.\*;

public class Receiver

{

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

{

InetAddress addr = InetAddress.getByName("192.168.117.185");

Socket s = new Socket(addr, 1234);

DataInputStream dis = new DataInputStream(s.getInputStream());

DataOutputStream dos = new DataOutputStream(s.getOutputStream());

int n = dis.readInt();

System.out.println("N = "+n);

int sum = 0;

int curr;

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

{

curr = dis.readInt();

System.out.println("element = "+curr);

sum += curr;

}

int check = dis.readInt();

System.out.println("check = "+check);

sum += check;

System.out.println("CheckSum is : "+sum);

int res = sum & (sum+1);

if(res == 0)

System.out.println("Valid");

else

System.out.println("Invalid");

}

}

**SAMPLE INPUT AND OUTPUT :**

|  |  |
| --- | --- |
| java Sender:  Enter no. of data :  5  1  2  3  4  5  Checksum Calculated : -16 | java Reciever:  N = 5  element = 1  element = 2  element = 3  element = 4  element = 5  check = -16  CheckSum is : -1  Valid |

**PROGRAM :**

**LinkState.cpp**

#include<bits/stdc++.h>

using namespace std;

# define INF 0x3f3f3f3f

typedef pair<int, int> iPair;

class Graph

{

int V;

list< pair<int, int> > \*adj;

public:

Graph(int V);

void addEdge(int u, int v, int w);

void shortestPath(int s);

};

Graph::Graph(int V)

{

this->V = V;

adj = new list<iPair> [V];

}

void Graph::addEdge(int u, int v, int w)

{

adj[u].push\_back(make\_pair(v, w));

adj[v].push\_back(make\_pair(u, w));

}

void Graph::shortestPath(int src)

{

priority\_queue< iPair, vector <iPair> , greater<iPair> > pq;

vector<int> dist(V, INF);

pq.push(make\_pair(0, src));

dist[src] = 0;

while (!pq.empty())

{

int u = pq.top().second;

pq.pop();

list< pair<int, int> >::iterator i;

for (i = adj[u].begin(); i != adj[u].end(); ++i)

{

int v = (\*i).first;

int weight = (\*i).second;

if (dist[v] > dist[u] + weight)

{

dist[v] = dist[u] + weight;

pq.push(make\_pair(dist[v], v));

}

}

}

printf("Vertex Distance from Source\n");

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

printf("%d \t\t %d\n", i, dist[i]);

}

int main()

{

int V = 5;

Graph g(V);

g.addEdge(0, 1, 4);

g.addEdge(0, 3, 7);

g.addEdge(1, 3, 1);

g.addEdge(1, 2, 2);

g.addEdge(2, 3, 5);

g.addEdge(2, 4, 8);

g.addEdge(3, 4, 6);

for(int i=0; i<5; i++)

{

cout<<"Source vertex is:"<<i<<endl;

g.shortestPath(i);

cout<<endl;

}

return 0;

}

**SAMPLE GRAPH :**

**SAMPLE INPUT AND OUTPUT :**

Source vertex is:0

Vertex Distance from Source

0 0

1 4

2 6

3 5

4 11

Source vertex is:1

Vertex Distance from Source

0 4

1 0

2 2

3 1

4 7

Source vertex is:2

Vertex Distance from Source

0 6

1 2

2 0

3 3

4 8

Source vertex is:3

Vertex Distance from Source

0 5

1 1

2 3

3 0

4 6

Source vertex is:4

Vertex Distance from Source

0 11

1 7

2 8

3 6

4 0

**PROGRAM :**

**DistanceVector.cpp**

    #include <iostream>

    #include <stdio.h>

    #include <limits.h>

    using namespace std;

    struct Edge

    {

        int src, dest, weight;

    };

    struct Graph

    {

        int V, E;

        struct Edge\* edge;

    };

    struct Graph\* createGraph(int V, int E)

    {

        struct Graph\* graph = new Graph;

        graph->V = V;

        graph->E = E;

        graph->edge = new Edge[E];

        return graph;

    }

    void printArr(int dist[], int n)

    {

        printf("Vertex Distance from Source\n");

        for (int i = 0; i < n; ++i)

        {

            printf("%d \t\t ", i);

            if(dist[i] == INT\_MAX)

            {

                printf("Cannot be reached\n");

            }

            else

            {

                printf("%d\n",dist[i]);

            }

        }

    }

    struct Graph\* addEdge(int sour, int desti, int w, struct Graph\* graph)

    {

        static int ct=0;

        graph->edge[ct].src = sour;

        graph->edge[ct].dest = desti;

        graph->edge[ct].weight = w;

        ct++;

        graph->edge[ct].src = desti;

        graph->edge[ct].dest = sour;

        graph->edge[ct].weight = w;

        ct++;

        return graph;

    }

    void BellmanFord(struct Graph\* graph, int src)

    {

        int V = graph->V;

        int E = graph->E;

        int dist[V];

        for (int i = 0; i < V; i++)

            dist[i] = INT\_MAX;

        dist[src] = 0;

        for (int i = 1; i <= V-1; i++)

        {

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

            {

                int u = graph->edge[j].src;

                int v = graph->edge[j].dest;

                int weight = graph->edge[j].weight;

                if (dist[u] != INT\_MAX && dist[u] + weight < dist[v])

                    dist[v] = dist[u] + weight;

            }

        }

        for (int i = 0; i < E; i++)

        {

            int u = graph->edge[i].src;

            int v = graph->edge[i].dest;

            int weight = graph->edge[i].weight;

            if (dist[u] != INT\_MAX && dist[u] + weight < dist[v])

            printf("Graph contains negative weight cycle");

        }

        printArr(dist, V);

        return;

    }

    int main()

    {

        int V = 5;

        int E = 14;

        struct Graph\* graph = createGraph(V, E);

        graph = addEdge(0, 1, 4 ,graph);

        graph = addEdge(0, 3, 7 ,graph);

        graph = addEdge(1, 3, 1 ,graph);

        graph = addEdge(1, 2, 2 ,graph);

        graph = addEdge(2, 3, 5 ,graph);

        graph = addEdge(2, 4, 8 ,graph);

        graph = addEdge(3, 4, 6 ,graph);

        for(int i=0;i<V;i++)

        {

            cout<<"Source is node "<<i<<endl;

            BellmanFord(graph, i);

            cout<<endl;

        }

        return 0;

    }

**SAMPLE GRAPH :**

**SAMPLE INPUT AND OUTPUT :**

Source vertex is:0

Vertex Distance from Source

0 0

1 4

2 6

3 5

4 11

Source vertex is:1

Vertex Distance from Source

0 4

1 0

2 2

3 1

4 7

Source vertex is:2

Vertex Distance from Source

0 6

1 2

2 0

3 3

4 8

Source vertex is:3

Vertex Distance from Source

0 5

1 1

2 3

3 0

4 6

Source vertex is:4

Vertex Distance from Source

0 11

1 7

2 8

3 6

4 0

**PROGRAM :**

**Ping.java**

import java.io.\*;

import java.net.\*;

class Ping

{

public static void sendPingRequest(String ipAddress) throws UnknownHostException, IOException

{

InetAddress geek = InetAddress.getByName(ipAddress);

System.out.println("Sending Ping Request to " + ipAddress);

if (geek.isReachable(5000))

System.out.println("Host is reachable");

else

System.out.println("Sorry ! We can't reach to this host");

}

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

{

String ipAddress = "127.0.0.1";

sendPingRequest(ipAddress);

ipAddress = "133.192.31.42";

sendPingRequest(ipAddress);

ipAddress = "192.168.0.102";

sendPingRequest(ipAddress);

}

}

**SAMPLE INPUT AND OUTPUT :**

Sending Ping Request to 127.0.0.1

Host is reachable

Sending Ping Request to 133.192.31.42

Sorry ! We can't reach to this host

Sending Ping Request to 192.168.0.102

Host is reachable

**PROGRAM :**

**Server.java**

import java.net.\*;

import java.io.\*;

public class Server

{

private Socket socket = null;

private ServerSocket server = null;

private DataInputStream in = null;

public Server(int port)

{

try

{

server = new ServerSocket(port);

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

System.out.println("Waiting for a client ...");

socket = server.accept();

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

in = new DataInputStream(new BufferedInputStream(socket.getInputStream()));

String line1 = "";

String line2 = "";

try

{

line1 = in.readUTF();

line2 = in.readUTF();

System.out.println("Data Recieved");

String ans = modDiv(line2,line1);

int fl = 0;

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

if(ans.substring(i,i+1).equals("1"))

{

fl = 1;

break;

}

}

if(fl == 1)

System.out.println("Incorrect data recieved");

else

System.out.println("Correct data recieved");

}

catch(IOException i)

{

System.out.println(i);

}

System.out.println("Closing connection");

socket.close();

in.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static boolean bitOf(char in) {

return (in == '1');

}

public static char charOf(boolean in) {

return (in) ? '1' : '0';

}

public String XOR(String a,String b){

System.out.println(a + " and "+ b);

StringBuilder sb = new StringBuilder();

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

{

sb.append(charOf(bitOf(a.charAt(i)) ^ bitOf(b.charAt(i))));

}

String result = sb.toString();

return result;

}

public String modDiv(String append\_str,String key)

{

int len = key.length();

int index = 0;

int len1 = len;

String temp = append\_str.substring(0,len);

System.out.println(append\_str);

while(len < append\_str.length())

{

if(temp.substring(0,1).equals("1"))

{

temp = XOR(key,temp) + append\_str.substring(len,len+1);

}

else

{

String temp2 = "";

for(int i = 0;i<len1;i++)

temp2+="0";

temp = XOR(temp2,temp) + append\_str.substring(len,len+1);

}

len+=1;

temp = temp.substring(1,temp.length());

}

if(temp.substring(0,1).equals("1"))

{

temp = XOR(key,temp);

}

else

{

String temp2 = "";

for(int i = 0;i<len1;i++)

{

temp2+="0";

}

temp = XOR(temp2,temp);

}

temp = temp.substring(1,temp.length());

return temp;

}

public static void main(String args[])

{

Server server = new Server(5000);

}

}

**Client.java**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class Client

{

private Socket socket = null;

private DataInputStream input = null;

private DataOutputStream out = null;

public Client(String address, int port)

{

try

{

socket = new Socket(address, port);

System.out.println("Connected");

input = new DataInputStream(System.in);

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

}

catch(Exception e)

{

System.out.println(e);

}

Scanner sc = new Scanner(System.in);

String line = "";

String divi;

System.out.println("Enter dividend");

divi = sc.nextLine();

String div;

System.out.println("Enter divisor");

div = sc.nextLine();

String temp = encodedData(divi,div);

try

{

System.out.println("Sending data");

System.out.println("key " + div);

System.out.println("Encoded data " + temp);

out.writeUTF(div);

out.writeUTF(temp);

System.out.println("Data sent");

}

catch(IOException i)

{

System.out.println(i);

}

try

{

input.close();

out.close();

socket.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static boolean bitOf(char in)

{

return (in == '1');

}

public static char charOf(boolean in)

{

return (in) ? '1' : '0';

}

public String XOR(String a,String b)

{

System.out.println(a + " and "+ b);

StringBuilder sb = new StringBuilder();

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

{

sb.append(charOf(bitOf(a.charAt(i)) ^ bitOf(b.charAt(i))));

}

String result = sb.toString();

return result;

}

public String modDiv(String append\_str,String key)

{

int len = key.length();

int index = 0;

int len1 = len;

String temp = append\_str.substring(0,len);

System.out.println(append\_str);

while(len < append\_str.length())

{

if(temp.substring(0,1).equals("1"))

{

temp = XOR(key,temp) + append\_str.substring(len,len+1);

}

else

{

String temp2 = "";

for(int i = 0;i<len1;i++)

temp2+="0";

temp = XOR(temp2,temp) + append\_str.substring(len,len+1);

}

len+=1;

temp = temp.substring(1,temp.length());

}

if(temp.substring(0,1).equals("1"))

{

temp = XOR(key,temp);

}

else

{

String temp2 = "";

for(int i = 0;i<len1;i++)

{

temp2+="0";

}

temp = XOR(temp2,temp);

}

temp = temp.substring(1,temp.length());

return temp;

}

public String encodedData(String data,String key)

{

int len = key.length();

String append\_str = data;

for(int i =1;i<len;i++)

{

append\_str+="0";

}

String rem = modDiv(append\_str,key);

String codeWord = data + rem;

return codeWord;

}

public static void main(String args[])

{

Client client = new Client("127.0.0.1", 5000);

}

}

**SAMPLE INPUT AND OUTPUT :**

|  |  |
| --- | --- |
| Java Server:  Server started  Waiting for a client ...  Client accepted  Data Recieved  100100001  1101 and 1001  1101 and 1000  1101 and 1010  1101 and 1110  0000 and 0110  1101 and 1101  Correct data recieved  Closing connection | Java Client:  Connected  Enter dividend  100100  Enter divisor  1101  100100000  1101 and 1001  1101 and 1000  1101 and 1010  1101 and 1110  0000 and 0110  1101 and 1100  Sending data  key 1101  Encoded data 100100001  Data sent |