LAB FILE

JAMIA MILLIA ISLAMIA

DEPARTMENT OF COMPUTER ENGINEERING

B.TECH(COMPUTER ENGINEERING)

SEMESTER-5th

Session: 2024-25

COMPUTER NETWORK

CEN-593

Submitted To:

Prof. Mohd Amjad

Dr. Musheer Ahmad

Dr. Hannan Mansoor

Submitted By:

MOHD HAMMAD QADIR

22BCS029

S.No	Question	Remarks	Date
1	WAP to implement Caeser Cipher. (Substitution Cipher)		09-08-2024
2	WAP to implement Rail Fence Cipher. (Transposition Cipher)		21-08-2024
3	WAP to implement Playfair Cipher .		28-08-2024
4	WAP to implement Vigenere Cipher.		16-10-2024
5	WAP to implement basic Sockets.		13-11-2024
6	Write a socket program for client-server, the client will send stream of 0's and 1's & the server will count number of 0's and 1's sent by client.		
7	WAP to implement Hill Cipher.		
8	Write a socket program to implement TCP client-server such that it can count number of files in a folder.		
9	Write a socket program such that client should be able to send the text and server checks that the received number of characters are in text.		
10	Write a socket program to implement multi-client system where server can stop particular words.		
11	WAP to implement RSA algorithm.		
12	WAP to implement Vernam Cipher.		

Q1) WAP to implement Caeser Cipher. (Substitution Cipher)#include <bits/stdc++.h>

```
using namespace std;


string message;
int key;
void input()
{
   cout << "Enter message : ";
   cin >> message;
   cout << "Enter key : ";
   cin >> key;
   key = key % 61;
}
void encrypt()
{
   input();
   for (int i = 0; i < message.length(); i++)
   {

      if (isalnum(message[i]))
```

```
{
    if (isupper(message[i]))
    {
        if (message[i] + key > 'Z')
        {
            message[i] = '0' + message[i] + key - 'Z' - 1;
        }
        else
            message[i] = 'A' + (message[i] - 'A' + key) % 26;
    }
    else if (islower(message[i]))
    {
        if (message[i] + key > 'z')
        {
            message[i] = 'A' + message[i] + key - 'z' - 1;
        }
        else
            message[i] = 'a' + (message[i] - 'a' + key) % 26;
    }
    else if (isdigit(message[i]))
    {
        if (message[i] + key > '9')
```

```
            {
                message[i] = 'a' + message[i] + key - '9' - 1;
            }
            else
                message[i] = '0' + (message[i] - '0' + key) % 10;
        }
    }
    cout <<"Encrypted message: " <<message << endl;
}
void decrypt()
{
    input();
    for (int i = 0; i < message.length(); i++)
    {
        if (isalnum(message[i]))
        {
            if (isupper(message[i]))
            {
                if (message[i] - key < 'A')
                {
                    message[i] = 'z' - ('A' - (message[i] - key)) + 1;
```

```
    }
    else
        message[i] = 'A' + (message[i] - 'A' - key) % 26;
}
else if (islower(message[i]))
{
    if (message[i] - key < 'a')
    {
        message[i] = '9' - ('a' - (message[i] - key)) + 1;
    }
    else
        message[i] = 'a' + (message[i] - 'a' - key) % 26;
}
else if (isdigit(message[i]))
{
    if (message[i] - key < '0')
    {
        message[i] = 'Z' - ('0' - (message[i] - key)) + 1;
    }
    else
        message[i] = '0' + (message[i] - '0' - key) % 10;
}
```

```
        }
    }
    cout <<"decrypted message: " <<message << endl;
}

int main()
{
    while (true)
    {

        cout << "E to Encrypt \nD to Decrypt \nExit \n";
        char choice[50];
        cin >> choice;
        if (!strcmp(choice, "E"))
        {
            encrypt();
        }
        else if (!strcmp(choice, "D"))
        {
            decrypt();
        }
        else if (!strcmp(choice, "E"))
```

```
    {
        exit(0);
    }
  }

  return 0;
}
```

OUTPUT

```
E to Encrypt
D to Decrypt
Exit
E
Enter message : Hammad
Enter key : 6
Encrypted message: Ngssgj
E to Encrypt
D to Decrypt
Exit
D
Enter message : Ngssgj
Enter key : 6
decrypted message: Hammad
E to Encrypt
D to Decrypt
Exit
```

Q 2)

```
#include <bits/stdc++.h>
using namespace std;

string message;
int key;

void input()
{
    cout << "Enter message: ";
    cin.ignore();
    getline(cin, message);

    cout << "Enter key: ";
    cin >> key;
    if (key == 1)
        return;
}

void encrypt()
{
```

```
    input();
    vector<vector<char>> matrix(key, vector<char>(message.length(),
'\n'));
    string result = "";

    int row = 0;
    bool downward = false;
    for (int j = 0; j < message.length(); j++)
    {
        if (row == 0 || row == key - 1)
        {
            downward = !downward;
        }
        matrix[row][j] = message[j];

        downward ? row++ : row--;
    }

    for (int r = 0; r < key; r++)
    {
        for (int j = 0; j < message.length(); j++)
        {
```

```
            if (matrix[r][j] != '\n')
            {
                result.push_back(matrix[r][j]);
            }
        }
    }

    cout << "Encrypt message: " << result << endl;
}

void decrypt()
{
    input();
    vector<vector<char>> matrix(key, vector<char>(message.length(),
'\n'));
    string result = "";

    int row = 0;
    bool downward = false;
    for (int j = 0; j < message.length(); j++)
    {
        if (row == 0)
```

```
        downward = true;
    if (row == key - 1)
        downward = false;
    matrix[row][j] = '*';
    downward ? row++ : row--;
}

int index = 0;
for (int i = 0; i < key; i++)
{
    for (int j = 0; j < message.length(); j++)
    {
        if (matrix[i][j] == '*' && index < message.length())
            matrix[i][j] = message[index++];
    }
}

row = 0;
downward = false;
for (int j = 0; j < message.length(); j++)
{
    if (row == 0)
```

```
            downward = true;
        if (row == key - 1)
            downward = false;
        if (matrix[row][j] != '*')
            result.push_back(matrix[row][j]);
        downward ? row++ : row--;
    }

    cout << "Decrypt message: " << result << endl;
}

int main()
{



    while (true)
    {
        cout << "E to Encrypt \nD to Decrypt \nExit \n";
        string choice;
        cin >> choice;

        if (choice == "E")
```

```
        {
            encrypt();
        }
        else if (choice == "D")
        {
            decrypt();
        }
        else if (choice == "Exit")
        {
            break;
        }
    }

    return 0;
}
```

OUTPUT

```
E to Encrypt
D to Decrypt
Exit
E
Enter message: Hammad
Enter key: 3
Encrypt message: Haamdm
E to Encrypt
D to Decrypt
Exit
D
Enter message: Haamdm
Enter key: 3
Decrypt message: Hammad
E to Encrypt
D to Decrypt
Exit
```

Q3) WAP to implement Playfair Cipher .

```
#include <bits/stdc++.h>
using namespace std;
#define SIZE 30

void toLowerCase(string& str) {
    for (int i = 0; i < str.length(); i++) {
```

```
        if (str[i] > 64 && str[i] < 91)
            str[i] += 32;
    }
}

void removeSpaces(string& str) {
    str.erase(remove(str.begin(), str.end(), ' '), str.end());
}

void generateKeyTable(string key, char keyT[5][5]) {
    int i, j, k;
    int dicty[26] = { 0 };

    for (i = 0; i < key.length(); i++) {
        if (key[i] != 'j')
            dicty[key[i] - 97] = 2;
    }
    dicty['j' - 97] = 1;

    i = 0;
    j = 0;
    for (k = 0; k < key.length(); k++) {
```

```
    if (dicty[key[k] - 97] == 2) {
        dicty[key[k] - 97] -= 1;
        keyT[i][j] = key[k];
        j++;
        if (j == 5) {
            i++;
            j = 0;
        }
    }
}

for (k = 0; k < 26; k++) {
    if (dicty[k] == 0) {
        keyT[i][j] = (char)(k + 97);
        j++;
        if (j == 5) {
            i++;
            j = 0;
        }
    }
}
cout << "The Key Table is :" << endl;
```

```
    cout << "-----------------------" << endl;
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            cout << keyT[i][j] << " ";
        }
        cout << endl;
    }
}

void search(char keyT[5][5], char a, char b, int arr[]) {
    if (a == 'j')
        a = 'i';
    else if (b == 'j')
        b = 'i';

    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            if (keyT[i][j] == a) {
                arr[0] = i;
                arr[1] = j;
            } else if (keyT[i][j] == b) {
                arr[2] = i;
```

```
                arr[3] = j;
            }
        }
    }
}

int mod5(int a) { return (a % 5); }

void prepare(string& str) {
    if (str.length() % 2 != 0) {
        str += 'z';
    }
}

void encrypt(string& str, char keyT[5][5]) {
    int a[4];
    for (int i = 0; i < str.length(); i += 2) {
        search(keyT, str[i], str[i + 1], a);

        if (a[0] == a[2]) {
            str[i] = keyT[a[0]][mod5(a[1] + 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
```

```
        } else if (a[1] == a[3]) {
            str[i] = keyT[mod5(a[0] + 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
        } else {
            str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}

void decrypt(string& str, char keyT[5][5]) {
    int a[4];
    for (int i = 0; i < str.length(); i += 2) {
        search(keyT, str[i], str[i + 1], a);

        if (a[0] == a[2]) {
            str[i] = keyT[a[0]][mod5(a[1] + 4)];
            str[i + 1] = keyT[a[0]][mod5(a[3] + 4)];
        } else if (a[1] == a[3]) {
            str[i] = keyT[mod5(a[0] + 4)][a[1]];
            str[i + 1] = keyT[mod5(a[2] + 4)][a[1]];
        } else {
```

```
            str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}

void encryptByPlayfairCipher(string& str, string& key) {
    char keyT[5][5];
    toLowerCase(key);
    toLowerCase(str);

    removeSpaces(key);
    removeSpaces(str);

    prepare(str);
    generateKeyTable(key, keyT);
    encrypt(str, keyT);
}

void decryptByPlayfairCipher(string& str, string& key) {
    char keyT[5][5];
    toLowerCase(key);
```

```
    removeSpaces(key);
    generateKeyTable(key, keyT);
    decrypt(str, keyT);
}

int main() {
    string str, key;
    cout << "Enter key: ";
    getline(cin, key);

    cout << "Enter message: ";
    getline(cin, str);

    encryptByPlayfairCipher(str, key);
    cout << "Cipher text: " << str << endl;

    decryptByPlayfairCipher(str, key);
    cout << "Decrypted text: " << str << endl;

    return 0;
```

}

OUTPUT

```
Enter key: 3
Enter message: Hammad
The Key Table is :
------------------------
3 a b c d
e f g h i
k l m n o
p q r s t
u v w x y
Cipher text: fclbb3
The Key Table is :
------------------------
3 a b c d
e f g h i
k l m n o
p q r s t
u v w x y
Decrypted text: hamaad
```

Q4) WAP to implement Vigenere Cipher.

```
#include <bits/stdc++.h>
using namespace std;
```

```
string generateKey(string str, string key)
{
    int x = str.size();
    for (int i = 0;; i++)
    {
        if (key.size() == x)
            break;
        key.push_back(key[i % key.size()]);
    }
    return key;
}

string cipherText(string str, string key)
{
    string result;
    for (int i = 0; i < str.size(); i++)
    {

        char x = ((str[i] - 'a') + (key[i] - 'a')) % 26;
        x += 'a';
        result.push_back(x);
    }
```

```
    return result;
}

string originalText(string result, string key)
{
    string orig_text;
    for (int i = 0; i < result.size(); i++)
    {
        char x = ((result[i] - 'a') - (key[i] - 'a') + 26) % 26;
        x += 'a';
        orig_text.push_back(x);
    }
    return orig_text;
}

int main()
{
    string str = "Hammadqadir";
    string keyword = "AYUSH";

    transform(str.begin(), str.end(), str.begin(), ::tolower);
```

```
    transform(keyword.begin(), keyword.end(), keyword.begin(),
::tolower);


    string key = generateKey(str, keyword);
    string result = cipherText(str, key);


    cout << "Ciphertext : " << result << "\n";
    cout << "Original/Decrypted Text : " << originalText(result, key) <<
"\n";


    return 0;
}
```

OUTPUT

```
Ciphertext : hygehdouvpr
Original/Decrypted Text : hammadqadir
```

Q5) WAP to implement basics of Socket.

CLIENT:

```
const net = require('net');
const readline = require('readline');


const serverIp = '192.168.50.242';
```

```
const rl = readline.createInterface({
  input: process.stdin,
  output: null
});

const socket = net.createConnection({ port: 2000 });

socket.on('connect', () => {


  socket.on('data', data => {
    console.log("enter messsage");
    console.log('Received from server:', data.toString('utf8'));
  });

  rl.on('line', (input) => {

    socket.write(input);
  });
});
```

SERVER:

```
const net = require('net');
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: null
});

const server = net.createServer(socket => {
  console.log('Client connected');
  socket.on('data', data => {
    console.log('Received from client: ', data.toString('utf8'));
  });
  socket.on('end', () => {
    console.log('Client disconnected');
  });
  rl.on('line', input => {
    socket.write(input);

  });
});
```

```
server.listen(2000, () => {
  console.log('Server is listening on port 2000');
});
```

OUTPUT

SERVER

```
Server is listening on port 2000
Client connected
Hello how are u Clien?
Received from client:  I am fine Server.
```

CLIENT

```
enter messsage
Received from server: Hello how are u Clien?
I am fine Server.
```

Q6) Write a socket program for client-server, the client will send stream of 0's and 1's & the server will count number of 0's and 1's sent by client.

CLIENT:

```
const net = require('net');
const readline = require('readline');
```

```
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

const socket = net.createConnection({ port: 2000 });

socket.on('connect', () => {
console.log("Connected to server!!!");
rl.question("Enter a stream of 0's and 1's: ", (data) => {
  socket.write(data);
  socket.end();
  rl.close();
});
  socket.on('data', data => {

    console.log('Received from server:', data.toString('utf8'));
 });

});
```

SERVER:

```
const net = require("net");
const readline = require("readline");
let zerosCount = 0;
let onesCount = 0;
let receivedData = "";

const rl = readline.createInterface({
  input: process.stdin,
  output: null,
});

const server = net.createServer((socket) => {
  console.log("Client connected");
  socket.on("data", (data) => {
    receivedData += data.toString("utf-8");
    for (const char of data.toString()) {
      if (char === "0") {
        zerosCount++;
      } else if (char === "1") {
        onesCount++;
```

```
    }
   }

   console.log("Received from client: ", data.toString("utf8"));
  });
  socket.on("end", () => {
   console.log("Client disconnected");
   console.log(`The Data send by the client is: ${receivedData}`);
   console.log(`Number of 0's received: ${zerosCount}`);
   console.log(`Number of 1's received: ${onesCount}`);

   zerosCount = 0;
   onesCount = 0;
   receivedData = "";
  });

  rl.on("line", (input) => {
   socket.write(input);
  });
});

server.listen(2000, () => {
```

```
  console.log("Server is listening on port 2000");
});
```

OUTPUT

CLIENT:

```
Connected to server!!!
Enter a stream of 0's and 1's: 01010111110100000
01010111110100000
```

SERVER;

```
Server is listening on port 2000
Client connected
Received from client:  01010111110100000
Client disconnected
The Data send by the client is: 01010111110100000
Number of 0's received: 9
Number of 1's received: 8
```

Q7) WAP to implement Hill Cipher.

```
#include<iostream>
#include<vector>

using namespace std;
```

```
string encrypt(string msg,string  key ){
    int len=msg.length();
    vector<vector<int>> matrix(len,vector<int>(len,0));
    vector<int> msgvector(len);
    vector<int> ansvector(len);
    string ans;

    int k=0;

    for(int i=0;i<len; i++){
        for(int j=0 ;j<len;j++){
            matrix[i][j]=key[k++]-'A';
        }
    }

    for(int i=0;i<len;i++){
        msgvector[i]=msg[i]-'A';
    }

    for(int i=0;i<len;i++){
        int res=0;
        for(int j=0;j<len;j++){
```

```
            res+=matrix[i][j]*msgvector[j];
        }
        res = (res % 26 + 26) % 26;
        ansvector[i]=res;
    }

    for(int i=0;i<len;i++){
        ans[i]=char(ansvector[i]+'A');
        cout<<ans[i];
    }

    return ans;
}


int modInverse(int a, int m) {
    a = (a % m + m) % m;
    for (int x = 1; x < m; x++) {
        if ((a * x) % m == 1) return x;
    }
    return -1; // No modular inverse exists
}
```

```
void getCof(vector<vector<int>>& mat, vector<vector<int>>& cof, int p,
int q, int n) {
    int i = 0, j = 0;
    for (int row = 0; row < n; row++) {
        for (int col = 0; col < n; col++) {
            if (row != p && col != q) {
                cof[i][j++] = mat[row][col];
                if (j == n - 1) {
                    j = 0;
                    i++;
                }
            }
        }
    }
}

int getDet(vector<vector<int>>& mat, int n) {
    if (n == 1) return mat[0][0];

    int det = 0;
    vector<vector<int>> cof(mat.size(), vector<int>(mat.size()));
```

```
    int sign = 1;
    for (int f = 0; f < n; f++) {
        getCof(mat, cof, 0, f, n);
        det += sign * mat[0][f] * getDet(cof, n - 1);
        sign = -sign;
    }
    return det;
}

void adjoint(vector<vector<int>>& mat, vector<vector<int>>& adj) {
    int n = mat.size();
    if (n == 1) {
        adj[0][0] = 1;
        return;
    }

    int sign = 1;
    vector<vector<int>> cof(n, vector<int>(n));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            getCof(mat, cof, i, j, n);
            sign = ((i + j) % 2 == 0) ? 1 : -1;
```

```
            adj[j][i] = sign * getDet(cof, n - 1); // Transpose and cofactor
        }
    }
}

bool inverse(vector<vector<int>>& mat, vector<vector<int>>& inv, int mod) {
    int n = mat.size();
    int det = getDet(mat, n);
    det = (det % mod + mod) % mod; // Ensure positive determinant
    int detInv = modInverse(det, mod);
    if (detInv == -1) {
        cout << "Singular matrix, can't find its inverse\n";
        return false;
    }

    vector<vector<int>> adj(n, vector<int>(n));
    adjoint(mat, adj);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            inv[i][j] = (adj[i][j] * detInv % mod + mod) % mod; // Modular inverse
```

```
        }
    }
    return true;
}

string decrypt(string msg, string key) {
    int len = msg.length();
    vector<vector<int>> matrix(len, vector<int>(len));
    vector<vector<int>> inversematrix(len, vector<int>(len));
    vector<int> msgvector(len);
    vector<int> ansvector(len);
    string ans(len, ' ');

    int k = 0;
    for (int i = 0; i < len; i++) {
        for (int j = 0; j < len; j++) {
            matrix[i][j] = key[k++] - 'A';
        }
    }

    for (int i = 0; i < len; i++) {
        msgvector[i] = msg[i] - 'A';
```

```
    }

    if (!inverse(matrix, inversematrix, 26)) {
        cout << "Error: Key matrix is not invertible.\n";
        return "";
    }

    for (int i = 0; i < len; i++) {
        int res = 0;
        for (int j = 0; j < len; j++) {
            res += inversematrix[i][j] * msgvector[j];
        }
        res = (res % 26 + 26) % 26;
        ansvector[i] = res;
    }

    for (int i = 0; i < len; i++) {
        ans[i] = char(ansvector[i] + 'A');
    }

    return ans;
}
```

```
int main (){
   while (true)
   {
      int choice,len;
      string msg,key;
      cout << "\nChoose any one of the following" << endl
          << "Press 1 to encrypt" << endl
          << "Press 2 to decrypt"<<endl;
      cin >> choice;
      if (choice == 1)
      {
         cout<<"Enter the msg to be encrypted\n";
         cin>>msg;
         len=msg.length();
         cout<<"Enter the key(size should be of "<<len*len<<")\n";
         cin>>key;

         string ans=encrypt(msg,key);
         cout<<ans;
      }
      else if (choice == 2)
```

```
        {

            cout<<"Enter the msg to be decrypted\n";
            cin>>msg;
            len=msg.length();
            cout<<"Enter the key(size should be of "<<len*len<<")\n";
            cin>>key;
            string ans=decrypt(msg,key);
            cout<<ans;
        }
        else
        {
            break;
        }
    }
}
```

OUTPUT:

```
Choose any one of the following
Press 1 to encrypt
Press 2 to decrypt
1
Enter the msg to be encrypted
ACT
Enter the key(size should be of 9)
GYBNQKURP
POH
Choose any one of the following
Press 1 to encrypt
Press 2 to decrypt
2
Enter the msg to be decrypted
POH
Enter the key(size should be of 9)
GYBNQKURP
ACT
```

Q8) Write a socket program to implement TCP client-server such that it can count number of files in a folder.

Server Side:

```
const net = require('net');

const fs = require('fs');
```

```
const path = require('path');


const server = net.createServer((socket) => {
  console.log('Client connected');
  socket.on('data', (data) => {
    const folderPath = data.toString().trim();
    if (fs.existsSync(folderPath) && fs.lstatSync(folderPath).isDirectory()) {
      fs.readdir(folderPath, (err, files) => {
        if (err) {
          socket.write('Error reading directory');
        } else {
          const fileCount = files.length;
          socket.write(`Number of files in folder: ${fileCount}`);
        }
      });
    } else {
      socket.write('Invalid folder path');
    }
  });
  socket.on('end', () => {
    console.log('Client disconnected');
  });
```

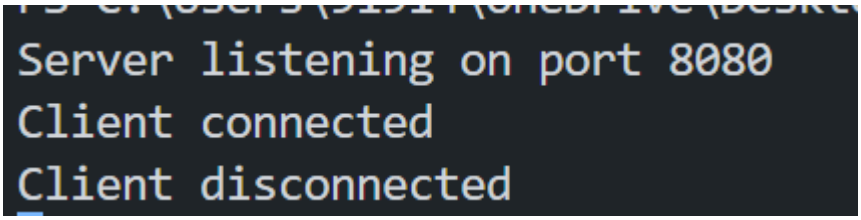
```
});
server.listen(8080, () => {
  console.log('Server listening on port 8080');
});
```

Client Side:

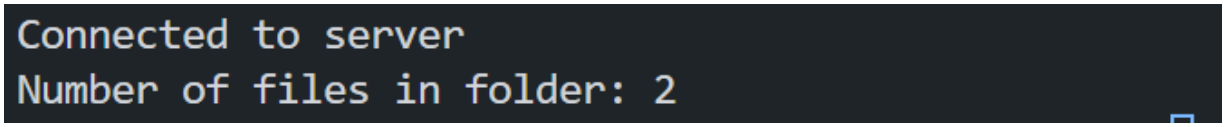
```
const net = require('net');
const client = net.createConnection({ port: 8080 }, () => {
console.log('Connected to server');

  const folderPath = 'C:\\Users\\91914\\OneDrive\\Desktop\\audio';
  client.write(folderPath);
});
client.on('data', (data) => {
  console.log(data.toString());
  client.end();
});

client.on('error', (err) => {
  console.error('Error:', err.message);
});
```

OUTPUT

Server Side:



Client Side:



Q9) Write a socket program such that client should be able to send the text and server checks that the received number of characters are in text.

Server Side:

```
const net = require('net');

const expectedLength = 10;
```

```
const server = net.createServer((socket) => {
  console.log('Client connected');

  socket.on('data', (data) => {
    const receivedText = data.toString().trim();

    if (receivedText.length === expectedLength) {
      socket.write(`Success: The number of characters is correct.And the message is ${receivedText}`);
    } else {
      socket.write(`Error: The received text should have ${expectedLength} characters. Received: ${receivedText.length}`);
    }
  });

  socket.on('end', () => {
    console.log('Client disconnected');
  });
});
const host = '192.168.15.107';
const port = 8080;
server.listen(port, host, () => {
  console.log(`Server listening on ${host}:${port}`);
```

```
});
```

Client Side:

```
const net = require('net');
const serverIp = '192.168.15.107';
const serverPort = 8080;

const client = net.createConnection({ host: serverIp, port: serverPort },
() => {
  console.log('Connected to server');
  const text = 'HellooWorld';
  client.write(text);
});

client.on('data', (data) => {
  console.log(data.toString());
  client.end(); se
});

client.on('error', (err) => {
  console.error('Error:', err.message);
```

});

OUTPUT

SERVER:

```
Server listening on 192.168.15.107:8080
Client connected
Client disconnected
```

CLIENT:

```
Connected to server
Error: The received text should have 10 characters. Received: 11
```

Q10) Write a socket program to implement multi-client system where server can stop particular words.

SERVER:

```
const net = require('net');

const forbiddenWords = ['badword', 'blocked'];
const server = net.createServer((socket) => {
  console.log('Client connected');
  socket.on('data', (data) => {
    let clientMessage = data.toString().trim();
```

```
    forbiddenWords.forEach(word => {
      const regex = new RegExp(word, 'gi');
      clientMessage = clientMessage.replace(regex, '***');
    });
    socket.write(`Modified message: ${clientMessage}`);
  });

  socket.on('end', () => {
    console.log('Client disconnected');
  });
});

server.listen(8080, () => {
  console.log('Server listening on port 8080');
});
```

CLIENT:

```
const net = require('net');

const serverIp = '192.168.15.107';
const serverPort = 8080;
```

```
const client = net.createConnection({ host: serverIp, port: serverPort },
() => {
  console.log('Connected to server');
  const message = 'This is a message with Badword in it.';
  client.write(message);
});

client.on('data', (data) => {
  console.log('Server response: ' + data.toString());
  client.end();
});

client.on('error', (err) => {
  console.error('Error:', err.message);
});
```

OUTPUT:

SERVER:

```
Server listening on port 8080
Client connected
Client disconnected
```

CLIENT:

```
Connected to server
Server response: Modified message: This is a message with *** in it.
```

Q11) WAP to implement RSA algorithm.

```
#include <bits/stdc++.h>
using namespace std;

// Function to calculate gcd
int gcd(int a, int b) {
    return (b == 0) ? a : gcd(b, a % b);
}

// Function to find modular inverse
int modInverse(int e, int phi) {
    for (int d = 1; d < phi; d++) {
        if ((e * d) % phi == 1) {
            return d;
        }
    }
    return -1;
}
```

```
// Fast modular exponentiation
long long modExpo(long long base, long long exp, long long mod) {
    long long result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1) {
            result = (result * base) % mod;
        }
        exp = exp >> 1;
        base = (base * base) % mod;
    }
    return result;
}

int main() {
    // Step 1: Key generation
    int p = 61; // First prime number
    int q = 53; // Second prime number
    int n = p * q; // Modulus
    int phi = (p - 1) * (q - 1); // Totient

    // Choose public key e
```

```
int e = 17; // e must be coprime to phi and 1 < e < phi
while (gcd(e, phi) != 1) {
   e++;
}

// Compute private key d
int d = modInverse(e, phi);

// Display keys
cout << "Public Key: (" << e << ", " << n << ")\n";
cout << "Private Key: (" << d << ", " << n << ")\n";

// Step 2: Encryption
int message;
cout << "Enter a message (integer): ";
cin >> message;

// Encrypt message
long long encrypted = modExpo(message, e, n);
cout << "Encrypted Message: " << encrypted << "\n";

// Step 3: Decryption
```

```
    long long decrypted = modExpo(encrypted, d, n);
    cout << "Decrypted Message: " << decrypted << "\n";

    return 0;
}
```

OUTPUT:

```
Public Key: (17, 3233)
Private Key: (2753, 3233)
Enter a message (integer): 45
Encrypted Message: 1086
Decrypted Message: 45
```

Q12) WAP to implement Vernam Cipher.

```
#include <iostream>
#include <string>
using namespace std;

// Function to encrypt/decrypt using Vernam Cipher
string vernamCipher(string text, string key) {
    string result = "";
    for (size_t i = 0; i < text.length(); i++) {
```

```
        // XOR operation between each character of the text and the key
        result += text[i] ^ key[i];
    }
    return result;
}

int main() {
    string plaintext, key;

    // Input plaintext
    cout << "Enter plaintext: ";
    getline(cin, plaintext);

    // Input key
    cout << "Enter key (same length as plaintext): ";
    getline(cin, key);

    // Check if the key and plaintext are the same length
    if (plaintext.length() != key.length()) {
        cout << "Error: Key length must be the same as plaintext length!"
<< endl;
        return 1;
```

```
    }

    // Encrypt the plaintext
    string ciphertext = vernamCipher(plaintext, key);
    cout << "Encrypted Ciphertext (in ASCII): ";
    for (char c : ciphertext) {
        cout << int(c) << " "; // Display ASCII values of the ciphertext
    }
    cout << endl;

    // Decrypt the ciphertext
    string decryptedText = vernamCipher(ciphertext, key);
    cout << "Decrypted Text: " << decryptedText << endl;

    return 0;
}
```

OUTPUT:

```
Enter plaintext: Hammad
Enter key (same length as plaintext): aurora
Encrypted Ciphertext (in ASCII): 41 20 31 2 19 5
Decrypted Text: Hammad
```