1. To calculate the hamming distance of the given codewords.

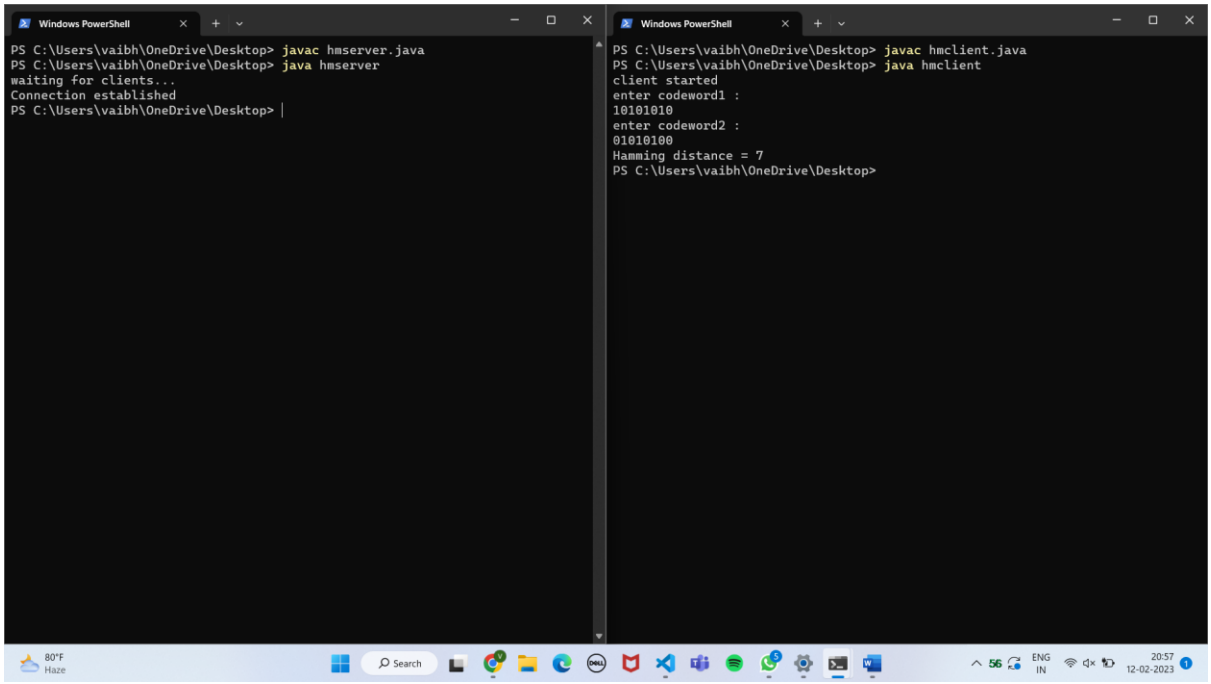
SERVER

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.*;
public class hmserver{
    static int calculatedistance(String num1, String num2){
        int count = 0;
        for(int i = 0;i<num1.length();i++){
            if(num1.charAt(i) != num2.charAt(i)){
                count++;
            }
            else{
                continue;
            }
        }
        return count;
    }
    public static void main(String [] args){
        try{
            System.out.println("waiting for clients...");
            ServerSocket ss = new ServerSocket(2380);
            Socket soc = ss.accept();
            System.out.println("Connection established");
            BufferedReader in = new BufferedReader(new
InputStreamReader(soc.getInputStream()));
            String s1 = in.readLine();
            String s2 = in.readLine();
            int n = calculatedistance(s1,s2);
            PrintWriter out = new PrintWriter(soc.getOutputStream(),true);
            out.println(n);
            ss.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

CLIENT

```java
import java.net.Socket;
import java.io.*;
public class hmclient{
    public static void main(String [] args){
        try{
            System.out.println("client started");
            Socket soc = new Socket("localhost", 2380);
            BufferedReader userinput = new BufferedReader(new
InputStreamReader(System.in));
            System.out.println("enter codeword1 : ");
            String s1 = userinput.readLine();
            System.out.println("enter codeword2 : ");
            String s2 = userinput.readLine();
            PrintWriter out = new PrintWriter(soc.getOutputStream(),true);
            out.println(s1);
            out.println(s2);
            BufferedReader in = new BufferedReader(new
InputStreamReader(soc.getInputStream()));
            System.out.println("Hamming distance =
"+Integer.parseInt(in.readLine()));
            soc.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

OUTPUT



2. To calculate the hamming distance of the given word.
   CODE

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
class hamming{
    public:
        string data; //it is the raw data received
        int m , r = 0; // n is the length of raw data and r is the number of
redundant bits
        char * msg; // it will store the all bits (data + redundant). We made
it dynamic because at compile time we dont know how much redundant bits will
be there, we will initialize memory to it once we know the number of redundant
bits.
        hamming(string data){
            this->data = data;
            //reversing the data received
            reverse(data.begin(),data.end());
            m = data.size();
            int power = 1;

            //finding the number of redundant bits and storing them in r
            while(power < (m + r + 1)){
                r++;
                power*=2;
            }
            //Allocating memory to our dynamic msg array(Note we are using one
based indexing).
            msg = new char[m+r+1];
            int curr = 0;

            //initializing the msg with data bits and for redundant bits, an
initial value of n
            for(int i = 1 ; i <= m+r ; i++){
                if(i & (i-1)){
                    msg[i] = data[curr++];
                }
                else msg[i] = 'n';
            }
            //function call to set the redundant bits
            setRedundantBits();
        }
        //function to show the whole msg
        void showmsg(){
            cout << "the data packet to be sent is : ";
            for(int i = m+r ; i >= 1 ; i--){
                cout << msg[i] << " ";
```

```
            }
            cout << endl;
        }
        void setRedundantBits(){
            int bit = 0;
            for(int i = 1 ; i <= m+r ; i*=2){
                int count = 0;
                for(int j = i+1 ; j<=m+r ; j++){
                    if(j & (1 << bit)){
                        if(msg[j] == '1') count++; // counting the number of
ones in corresponding data bits
                    }
                }
                if(count & 1) msg[i] = '1';
                else msg[i] = '0';
                bit++;
            }
            showmsg();
        }
        void receiver(){
            string ans = "";
            int bit = 0;
            for(int i = 1 ; i <= m+r ; i*=2){
                int count = 0;
                for(int j = i+1 ; j<=m+r ; j++){
                    if(j & (1 << bit)){
                        if(msg[j] == '1') count++;
                    }
                }
                if(count & 1){
                    if(msg[i] == '1') ans.push_back('0');
                    else ans.push_back('1');
                }
                else{
                    if(msg[i]=='0') ans.push_back('0');
                    else ans.push_back('1');
                }
                bit++;
            }
            if(ans.find('1') != string::npos){
                int power = 1;
                int wrongbit = 0;
                for(int i = 0 ; i < ans.size() ; i++){
                    if(ans[i]=='1') wrongbit+=power;
                    power*=2;
                }
                cout << "bit number " << wrongbit << " is wrong and having
error " << endl;
```

```
            }
            else{
                cout << "correct data packet received " << endl;
            }
        }
};
int main(){
    string data ;
    cin>>data;
    hamming h(data);
    h.receiver();
    return 0;
}
```

OUTPUT

```
PS C:\Users\vaibh> cd "c:\Users\vaibh\OneDrive\Desktop\" ; if ($?) { g
g } ; if ($?) { .\hamming }
1001101
the data packet to be sent is : 1 0 0 1 1 1 0 0 1 0 1
correct data packet received
PS C:\Users\vaibh\OneDrive\Desktop>
```