

CSE 1004

Network and Communication

LAB ASSESSMENT - 2

**NAME**: Vibhu Kumar Singh

**REG. NO**: 19BCE0215

**TEACHER**: Santhi H.

1. **Develop a menu-driven code to simulate the following error detecting and correction algorithms.**

**a) VRC**

**b) LRC**

**c) Checksum**

**d) CRC**

**e) Hamming Code**

**Ans 1)**

**Aim:** To simulate Error Detection algorithms in Networking.

The Detection Algorithms simulated are:

a) VRC

b) LRC

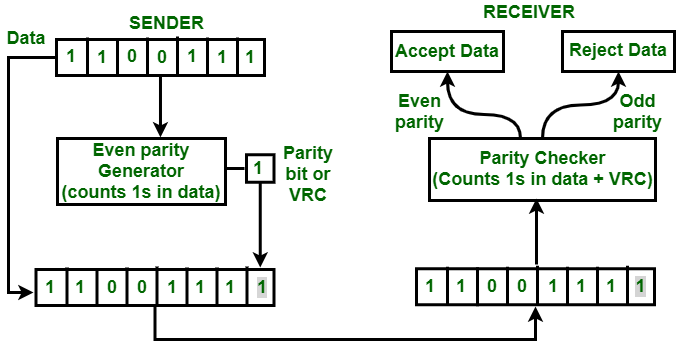
c) Checksum

d) CRC

e) Hamming Code

**Algorithms:**

1. **Vertical Redundancy Check (VRC):**



START:

* STEP 1: Get the input string from the user in binary (0 and 1).
* STEP 2: If the number of set bits is even, append 0. If the

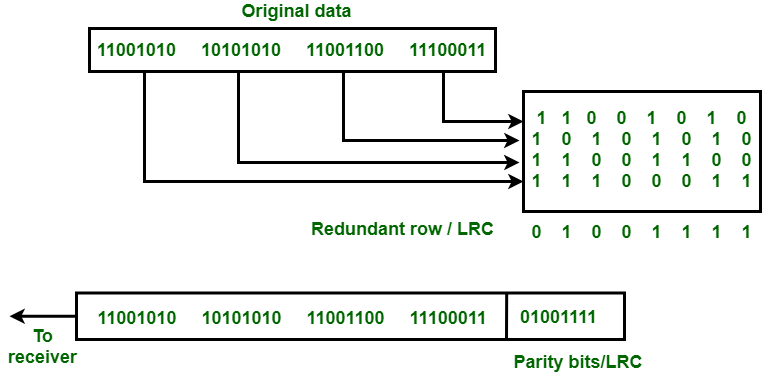
number of set bits are odd, append 1 in original string.

* STEP 3: This string message sent by the sender.
* STEP 4: Get the input for the message string received on the Receiver’s Side.
* STEP 5: If the Even-Parity Bit for the received string is 0, the

message has no error, else, it has error and is hence discarded.

END

1. **Longitudinal Redundancy Check (LRC):**



START:

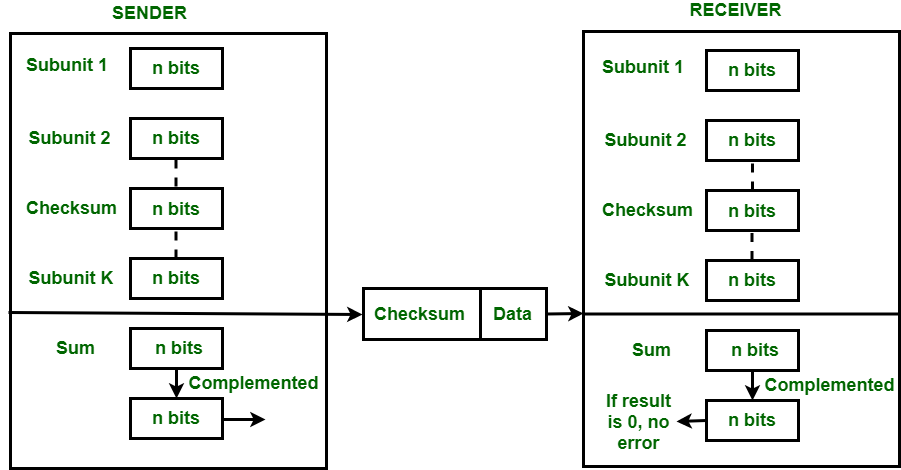
* STEP 1: Get the input matrix from the user in binary (0 and 1).

Let’s say the matrix entered is of dimensions (MXN).

* STEP 2: Compute and append the Even-Parity Bit for each row and column. IE: Dimensions are now (M+1)X(N+1).
* STEP 3: This string message sent by the sender in matrix form.
* STEP 4: Get the input for the message string received in matrix form on the Receiver’s Side.
* STEP 5: If the Even-Parity Bit for the matrix’s columns and rows is 0, the message has no error, else, it has error and the whole message block is discarded.

END

1. **Checksum:**

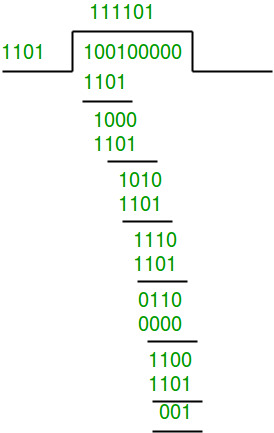
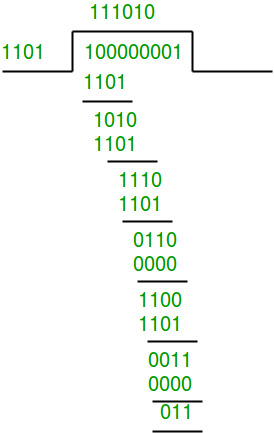


START:

* STEP 1: Choose for binary Checksum or Hexadecimal Checksum.
* STEP 2: Get the input message from the user (binary or characters).
* STEP 3: Convert the characters in hexadecimal and calculate the sum of all the hex values, then complement the sum to get the Checksum.
* STEP 4: If the input is binary, add all the binary inputs one by one and if there is end carry, add it to the LSB side. Complementing the sum (after carry) will give the Checksum.
* STEP 5: Enter the received input message (binary or characters). If the Checksum for the received message is 0, the message is accepted. Else, rejected.

END

1. **Cyclic Redundancy Check (CRC):**

Sender’s Side Receiver’s Side

START:

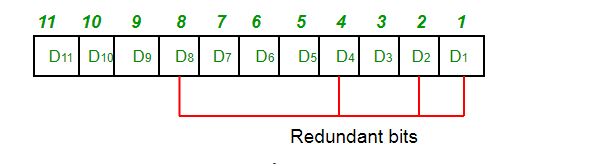
* STEP 1: Get the input string from the user in binary (0 and 1).

Get the key string from the user in binary (0 and 1).

* STEP 2: Append (length(key)-1) number of 0’s in the input string and take it as Dividend, while the key is the Divisor.
* STEP 3: Perform long division to calculate the Quotient and Remainder. Add the remainder to the Dividend and send the combined string to the Receiver Side.
* STEP 4: Get the input for the message string received in binary (0 and 1).
* STEP 5: Repeat the process of long division and if remainder is 0, accept the string. Else, reject it.

END

1. **Hamming Code:**



START:

* STEP 1: Get the input string from the user in binary (0 and 1).
* STEP 2: If the number of set bits is even, append 0. If the

number of set bits are odd, append 1 in original string.

* STEP 3: This is string message sent by the sender.
* STEP 4: Get the input for the message string received.
* STEP 5: If the Even-Parity Bit for the received string is 0, the

message has no error, else, it has error and is hence discarded.

END

**Menu-Driven Source Code:**

#include<bits/stdc++.h>

using namespace std;

string getParity(string str);

void crc\_receiver(char input[],char key[],int keylen,int msglen);

void vrc(string str)

{

    string p=getParity(str);

    cout<<"The Even-Parity Bit: "<<p<<"\n";

    string comb=p.append("|");

    comb=comb.append(str);

    cout<<"The combined message with Even-Parity Bit (at the starting): "<<comb<<"\n";

}

string getParity(string str)

{

    int n=str.length(),count=0;

    for(int i=0;i<n;i++)

    {

        if(str[i]=='1')

        {

            count++;

        }

    }

    string ParityBit;

    if(count%2==0)

    {

        ParityBit="0";

    }

    else

    {

        ParityBit="1";

    }

    return ParityBit;

}

void vrc\_main()

{

    start\_vrc:

    cout<<"\*\*\*SENDER'S SIDE\*\*\*\n\n";

    cout<<"The input message(string): ";

    string str\_vrc,str\_vrc\_rec;

    cin>>str\_vrc;

    for(int i=0;i<str\_vrc.length();i++)

    {

        if(!(str\_vrc[i]=='0'||str\_vrc[i]=='1'))

        {

            cout<<"Wrong input, Please try again.\n";

            system("pause");

            goto start\_vrc;

        }

    }

    vrc(str\_vrc);

    cout<<"\n\n\*\*\*RECEIVER'S SIDE\*\*\*\n\n";

    cout<<"Enter the message recieved(string): ";

    cin>>str\_vrc\_rec;

    if(getParity(str\_vrc\_rec)=="0")

    {

        cout<<"\nThere is NO error because the string contains EVEN number of set bits, the message is ACCEPTED.\n\n";

    }

    else

    {

        cout<<"\nThere is an error because the string contains ODD number of set bits, the message is REJECTED.\n\n";

    }

    system("pause");

}

void lrc()

{

    start\_lrc:

    int flag=0;

    cout<<"\*\*\*SENDER'S SIDE\*\*\*\n\n";

    datablock:

        cout<<"Enter the number of Data Blocks: ";

        int n;

        cin>>n;

        if(n<=0)

        {

            cout<<"\nPlease enter a positive value.\n";

            goto datablock;

        }

    databits:

        cout<<"Enter the number of bits per block: ";

        int m;

        cin>>m;

        if(m<=0)

        {

            cout<<"\nPlease enter a positive value.\n";

            goto databits;

        }

    matrix:

        int flagg=0;

        cout<<"Enter the Data Blocks (matrix form): \n";

        int arr[n+1][m+1];

        for(int i=0;i<n;i++)

        {

            for(int j=0;j<m;j++)

            {

                cin>>arr[i][j];

            }

        }

        for(int i=0;i<n;i++)

        {

            for(int j=0;j<m;j++)

            {

                if(arr[i][j]!=0 && arr[i][j]!=1)

                {

                    flagg=1;

                    break;

                }

            }

        }

        if(flagg==1)

        {

            cout<<"\nPlease enter binary input only.\n";

            goto matrix;

        }

    int XOR=0;

    for(int i=0;i<n;i++)

    {

        for(int j=0;j<m;j++)

        {

            XOR=XOR^arr[i][j];

        }

        arr[i][m]=XOR;

        XOR=0;

    }

    int XORR=0;

    for(int j=0;j<m+1;j++)

    {

        for(int i=0;i<n;i++)

        {

            XORR^=arr[i][j];

        }

        arr[n][j]=XORR;

        XORR=0;

    }

    cout<<"\nThe Matrix with Even Parity Bits: \n";

    for(int i=0;i<n+1;i++)

    {

        for(int j=0;j<m+1;j++)

        {

            cout<<arr[i][j]<<" ";

        }

        cout<<"\n";

    }

    cout<<"\nThe combined messsage sent is: ";

    for(int i=0;i<n+1;i++)

    {

        for(int j=0;j<m+1;j++)

        {

            cout<<arr[i][j];

        }

        cout<<" ";

    }

    cout<<"\n\n\*\*\*RECEIVER'S SIDE\*\*\*\n\n";

    cout<<"Enter the message received (matrix form): \n";

    int arr\_rec[n+1][m+1];

    for(int i=0;i<n+1;i++)

    {

        for(int j=0;j<m+1;j++)

        {

            cin>>arr\_rec[i][j];

        }

    }

    int XOR\_REC=0;

    for(int i=0;i<n+1;i++)

    {

        for(int j=0;j<m+1;j++)

        {

            XOR\_REC^=arr\_rec[i][j];

        }

//*cout<<XOR\_REC;*

        if(XOR\_REC!=0)

        {

            flag=1;

            break;

        }

    }

    if(flag==1)

    {

        cout<<"\nThere is an error because a row or column contains ODD number of set bits, the message is REJECTED.\n";

    }

    else

    {

        cout<<"\nThere is NO error becuase every row and column has EVEN number of set bits, the message is ACCEPTED.\n";

    }

    system("pause");

}

char data[100];

int rightSum(int l)

{

  int sum=0, i=1;

  for(;i<l;i=i+2)

    sum=sum + (int) data[i];

  return sum;

}

int leftSum(int l)

{

  int sum=0, i=0;

  for(;i<l;i=i+2)

    sum=sum + (int) data[i];

  return sum;

}

void checksum\_hex()

{

    char buf[100];

    int i, n, op=0, irs=0, ils=0, prs=0, cls=0, wc=0, pls=0, s=0, ocs=0, len=0;

    cout<<"\*\*\*\n\nSENDER'S SIDE\*\*\*\n\n";

    printf("Enter the data to be transmitted: ");

    gets(buf);

    gets(data);

    len=strlen(data);

    if(len%2!=0)

        len++;

    irs=rightSum(len);

    prs=irs%256;

    cls=irs/256;

    ils=cls+leftSum(len);

    pls=ils%256;

    wc=ils/256;

    s=pls\*256+prs+wc;

    ocs = 65535 - s;

    printf("The checksum generated is %X\n", ocs);

    char cs[100];

    int ch[100];

    cout<<"\n\nRECEIVER'S SIDE\*\*\*\n\n";

    printf("Enter the data received: ");

    gets(data);

    printf("Enter the received checksum: ");

    gets(cs);

    len=strlen(data);

    if(len%2!=0)

    len++;

    for(i=0;i<strlen(cs);i++)

    {

        if(cs[i]>='0' && cs[i]<='9')

        ch[i]=cs[i]-48;

        else if(cs[i]>='A' && cs[i]<='F')

        ch[i]=cs[i]-55;

        else if(cs[i]>='a' && cs[i]<='f')

        ch[i]=cs[i]-87;

    }

    irs=rightSum(len) + ch[2]\*16 + ch[3];

    prs=irs%256;

    cls=irs/256;

    ils=cls+leftSum(len) + ch[0]\*16 + ch[1];

    pls=ils%256;

    wc=ils/256;

    s=pls\*256+prs+wc;

    ocs = 65535 - s;

    if(ocs==0)

        printf("\nThe message is accepted!\n");

    else

        printf("\nThe message is rejected as there is an error!\n");

    system("pause");

}

string addBinary(string a, string b)

{

    string result = "";

    int s = 0;

    int i = a.size() - 1, j = b.size() - 1;

    while (i >= 0 || j >= 0 || s == 1)

    {

        s += ((i >= 0)? a[i] - '0': 0);

        s += ((j >= 0)? b[j] - '0': 0);

        result = char(s % 2 + '0') + result;

        s /= 2;

        i--; j--;

    }

    return result;

}

void checksum\_binary()

{

    int n;

    cout<<"\n\n\*\*\*SENDER'S SIDE\*\*\*";

    number:

    cout<<"\n\nEnter number of strings: ";

    cin>>n;

    if(n<=0 || isalpha(n))

    {

        cout<<"\nEnter a valid input (integer).\n";

        goto number;

    }

    string arr[n];

    for(int i=0;i<n;i++)

    {

        cout<<"Enter msg["<<i<<"]: ";

        cin>>arr[i];

    }

    string result;

    for(int i=0;i<n;i++)

    {

        result = addBinary(result,arr[i]);

    }

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    cout<<"Result= ";

    cout<<result<<endl;

    string carry;

    int len = arr[0].length();

    int result\_len = result.length();

    int c = result\_len-len;

    cout<<"Carry= ";

    if(c==1)

    {

        carry = result.at(0);

    }

    else

    {

        carry = result.substr(0,c);

    }

    string carryless = result.substr(c,len);

    cout<<carry<<endl;

    cout<<"Result without Carry = ";

    cout<<carryless<<endl;

    string sum = addBinary(carry,carryless);

    cout<<"Sum(Carry+Carryless) = ";

    cout<<sum<<endl;

    string checksum;

    for(int i=0;i<len;i++)

    {

        checksum = checksum+'0';

    }

    for(int i=0;i<len;i++)

    {

        if(sum[i] == '0')

        {

            checksum[i] = '1';

        }

    }

    cout<<"\nChecksum Generated= "<<checksum<<endl;

    cout<<"\n\n\*\*\*RECEIVER'S SIDE\*\*\*\n\n";

    cout<<"Enter the strings: \n";

    string arr\_rec[n+1];

    for(int i=0;i<n;i++)

    {

        cout<<"Enter msg["<<i<<"]: ";

        cin>>arr\_rec[i];

    }

    cout<<"Enter the Checksum received: ";

    string checksum\_inp;

    cin>>checksum\_inp;

    arr\_rec[n]=checksum\_inp;

    string result\_rec;

    for(int i=0;i<n+1;i++)

    {

        result\_rec=addBinary(result\_rec,arr\_rec[i]);

    }

    string carry\_rec;

    int len\_rec = arr\_rec[0].length();

    int result\_len\_rec = result\_rec.length();

    int c\_rec = result\_len\_rec-len\_rec;

    if(c\_rec==1)

    {

        carry\_rec= result\_rec.at(0);

    }

    else

    {

        carry\_rec= result\_rec.substr(0,c\_rec);

    }

    string carryless\_rec= result\_rec.substr(c,len);

    string sum\_rec = addBinary(carry\_rec,carryless\_rec);

    string checksum\_rec;

    for(int i=0;i<len\_rec;i++)

    {

        checksum\_rec=checksum\_rec+'0';

    }

    for(int i=0;i<len\_rec;i++)

    {

        if(sum\_rec[i] == '0')

        {

            checksum\_rec[i] = '1';

        }

    }

    cout<<"\nChecksum Generated= "<<checksum\_rec<<endl;

    int flaggg=0;

    for(int i=0;i<checksum\_rec.length();i++)

    {

        if(checksum\_rec[i]!=0)

        {

            flaggg=1;

            break;

        }

    }

    if(flaggg=1)

    {

        cout<<"\nThe Checksum is not 0, the message is rejected.\n";

    }

    else

    {

        cout<<"\nThe Checksum is 0, the message is accepted.\n";

    }

    system("pause");

}

void checksum()

{

    int choice;

    cout<<"1. Binary Checksum\n2. Hexadecimal Checksum\n0. Exit to Main Menu\nEnter your choice: ";

    cin>>choice;

    switch(choice)

    {

        case 0:

            return;

        break;

        case 1:

            checksum\_binary();

        break;

        case 2:

            checksum\_hex();

        break;

    }

}

void crc()

{

    start\_crc:

    cout<<"\*\*\*SENDER'S SIDE\*\*\*\n\n";

    message:

    int flag=0,flagg=0;

    int i,j,keylen,msglen;

    char input[100],key[30],temp[30],quot[100],rem[30],key1[30],temp1;

    printf("Enter the input message: ");

    scanf("%c",&temp1);

    gets(input);

    for(int i=0;i<strlen(input);i++)

    {

        if(input[i]!='0' && input[i]!='1')

        {

            flag=1;

            break;

        }

    }

    if(flag==1)

    {

        cout<<"\nPlease enter only binary input.\n";

        goto message;

    }

    key:

    printf("Enter the Key: ");

    gets(key);

    for(int i=0;i<strlen(key);i++)

    {

        if(key[i]!='0' && key[i]!='1')

        {

            flagg=1;

            break;

        }

    }

    if(flagg==1)

    {

        cout<<"\nPlease enter only binary input.\n";

        goto key;

    }

    keylen=strlen(key);

    msglen=strlen(input);

    strcpy(key1,key);

    for (i=0;i<keylen-1;i++)

    {

        input[msglen+i]='0';

    }

    for (i=0;i<keylen;i++)

     temp[i]=input[i];

    for (i=0;i<msglen;i++)

    {

        quot[i]=temp[0];

        if(quot[i]=='0')

             for (j=0;j<keylen;j++)

                key[j]='0';

        else

             for (j=0;j<keylen;j++)

                key[j]=key1[j];

        for (j=keylen-1;j>0;j--)

        {

            if(temp[j]==key[j])

                rem[j-1]='0';

            else

                rem[j-1]='1';

        }

        rem[keylen-1]=input[i+keylen];

        strcpy(temp,rem);

    }

    strcpy(rem,temp);

    printf("Quotient is: ");

    for (i=0;i<msglen;i++)

        printf("%c",quot[i]);

    printf("\nRemainder is: ");

    for (i=0;i<keylen-1;i++)

        printf("%c",rem[i]);

    printf("\nThe combined message sent is: ");

    for (i=0;i<msglen;i++)

        printf("%c",input[i]);

    for (i=0;i<keylen-1;i++)

        printf("%c",rem[i]);

    cout<<"\n\n\*\*\*RECEIVER'S SIDE\*\*\*\n\n";

    cout<<"Enter the message received: ";

    char crc\_rec[100];

    gets(crc\_rec);

    crc\_receiver(crc\_rec,key,msglen,keylen);

    cout<<"\n";

    system("pause");

}

void crc\_receiver(char input[],char key[],int msglen,int keylen)

{

    int flag=0,i=0;

    int j=0;

    char temp[100],rem[100],quot[100],key1[100];

    strcpy(key1,key);

    for(i=0;i<keylen;i++)

     temp[i]=input[i];

    for (i=0;i<msglen;i++)

    {

        quot[i]=temp[0];

        if(quot[i]=='0')

             for (j=0;j<keylen;j++)

                key[j]='0';

        else

             for (j=0;j<keylen;j++)

                key[j]=key1[j];

        for (j=keylen-1;j>0;j--)

        {

            if(temp[j]==key[j])

                rem[j-1]='0';

            else

                rem[j-1]='1';

        }

        rem[keylen-1]=input[i+keylen];

        strcpy(temp,rem);

    }

    strcpy(rem,temp);

    cout<<"Remainder: ";

    for(i=0;i<strlen(rem);i++)

    {

        cout<<rem[i];

    }

    for(i=0;i<strlen(rem);i++)

    {

        if(rem[i]!='0')

        {

            flag=1;

            break;

        }

    }

    if(flag==1)

    {

        cout<<"\n\nThere is an error as remainder is not 0, the message is REJECTED.\n";

    }

    else

    {

        cout<<"\n\nThere is NO error as remainder is 0, the message is ACCEPTED.\n";

    }

}

void hamming()

{

    start\_hamming:

    cout<<"\*\*\*SENDER'S SIDE\*\*\*\n\n";

    length:

    int maxp=6;

    int a[50],temp[70],temp2[70];

    int t,i,j,k,nd,n,nh,sum=0,pos=0;

    printf("Enter the length of input message: ");

    scanf("%d",&nd);

    if(nd<=0)

    {

        cout<<"\nPlease enter a positive value.\n";

        goto length;

    }

    message:

    int flag=0;

    printf("Enter the input message: ");

    for(i=0;i<nd;i++)

    {

        scanf("%d",&a[i]);

    }

    for(i=0;i<nd;i++)

    {

        if(a[i]!=1 && a[i]!=0)

        {

            flag=1;

        }

    }

    if(flag==1)

    {

        cout<<"\nPlease enter binary input only.\n";

        goto message;

    }

    for(i=0,j=0;i<nd;i++)

    {

        for(k=0;k<maxp;k++)

        {

            t=pow(2,k)-1;

            if(j==t)

            {

                temp[j]=0;

                j++;

            }

        }

        temp[j]=a[i];

        j++;

    }

    nh=j;

    printf("The length of Hamming Code: %d bits\n",nh);

    n=nh-nd;

    printf("The number of Parity Bits: %d \n",n);

    cout<<"---Calculating Parity Bits---\n";

    int b[n];

    int m=n-1;

    for(k=0;k<n;k++)

    {

        t=pow(2,k)-1;

        for(i=t;i<nh;)

        {

            for(j=0;j<=t;j++)

            {

                sum=sum+temp[i];

                i++;

                if(i>=nh)

                    break;

            }

            if(i>=nh)

                break;

            for(j=0;j<=t;j++)

            {

                i++;

                if(i>=nh)

                    break;

            }

            if(i>=nh)

                break;

        }

        temp[t]=sum%2;

        sum=0;

        printf("P%d: %d\n",t+1,temp[t]);

    }

    printf("\nHamming Code sent: ");

    for(i=0;i<nh;i++)

    {

        printf("%d ",temp[i]);

    }

    cout<<"\n\n\*\*\*RECEIVER'S SIDE\*\*\*\n\n";

   printf("Enter the Hamming Code received: ");

    for(i=0;i<nh;i++)

    {

        scanf("%d",&temp2[i]);

    }

    sum=0;

    for(k=0;k<n;k++)

    {

        t=pow(2,k)-1;

        for(i=t;i<nh;)

        {

            for(j=0;j<=t;j++)

            {

                sum=sum+temp2[i];

                i++;

                if(i>=nh)

                    break;

            }

            if(i>=nh)

                break;

            for(j=0;j<=t;j++)

            {

                i++;

                if(i>=nh)

                    break;

            }

            if(i>=nh)

                break;

        }

        b[m]=sum%2;

        sum=0;

        printf("P%d: %d\n",t+1,b[m]);

        m--;

    }

    for(m=0;m<n;m++)

    {

        pos=pos+b[n-m-1]\*pow(2,m);

    }

    printf("Position of Error: %d\n",pos);

    if(temp2[pos-1]==0)

        temp2[pos-1]=1;

    else

        temp2[pos-1]=0;

    printf("\nError Corrected: ");

    for(i=0;i<nh;i++)

    {

        printf("%d ",temp2[i]);

    }

    printf("\n",nd);

    system("pause");

}

int main()

{

    while(1)

    {

        start:

        system("cls");

        int choice;

        cout<<"Error Detection and Correction Algorithms (MENU): \n      [BY VIBHU KUMAR SINGH]\n\n1. Vertical Redundancy Check\n2. Longitudinal Redundancy Check \n3. Checksum\n4. Cyclic Redundancy Check \n5. Hamming Code \n0. Exit\nEnter your choice: ";

        cin>>choice;

        cout<<"<-------------------------------------------------------------------------------->\n\n";

        switch(choice)

        {

            case 1:

                vrc\_main();

            break;

            case 2:

                lrc();

            break;

            case 3:

            start\_checksum:

                checksum();

            break;

            case 4:

                crc();

            break;

            case 5:

                hamming();

            break;

            case 0:

                exit(0);

            break;

            default:

                cout<<"Invalid input";

                goto start;

        }

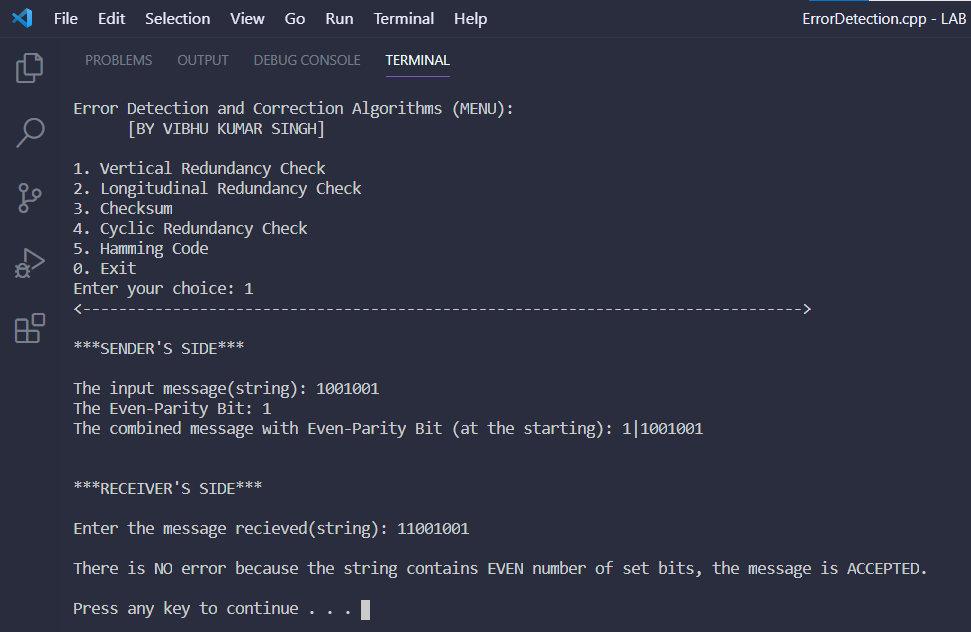
    }

}

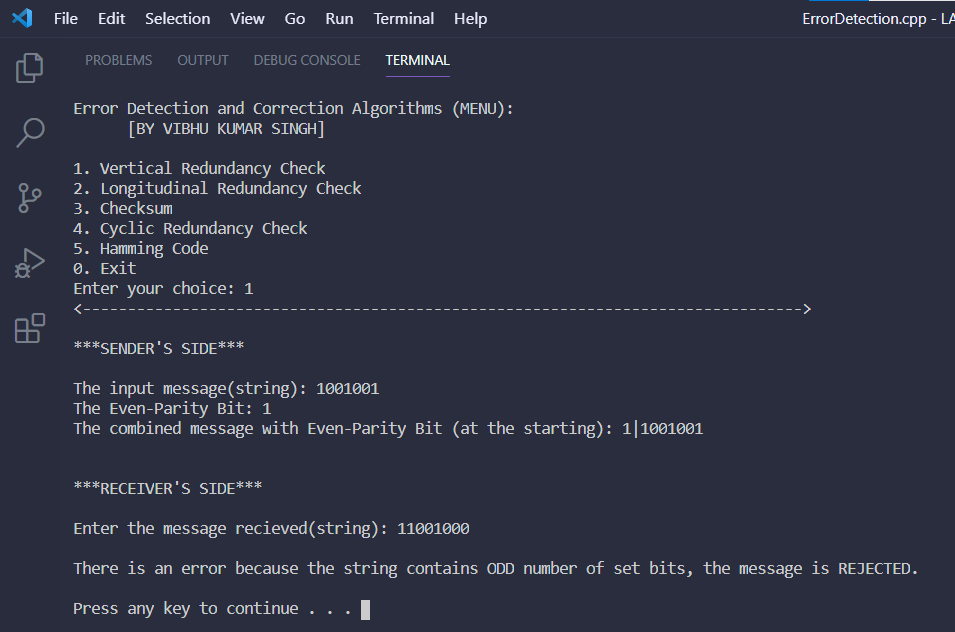
**OUTPUT SCREENSHOTS:**

1. **Vertical Redundancy Check (VRC):**

**No-Error Case:**

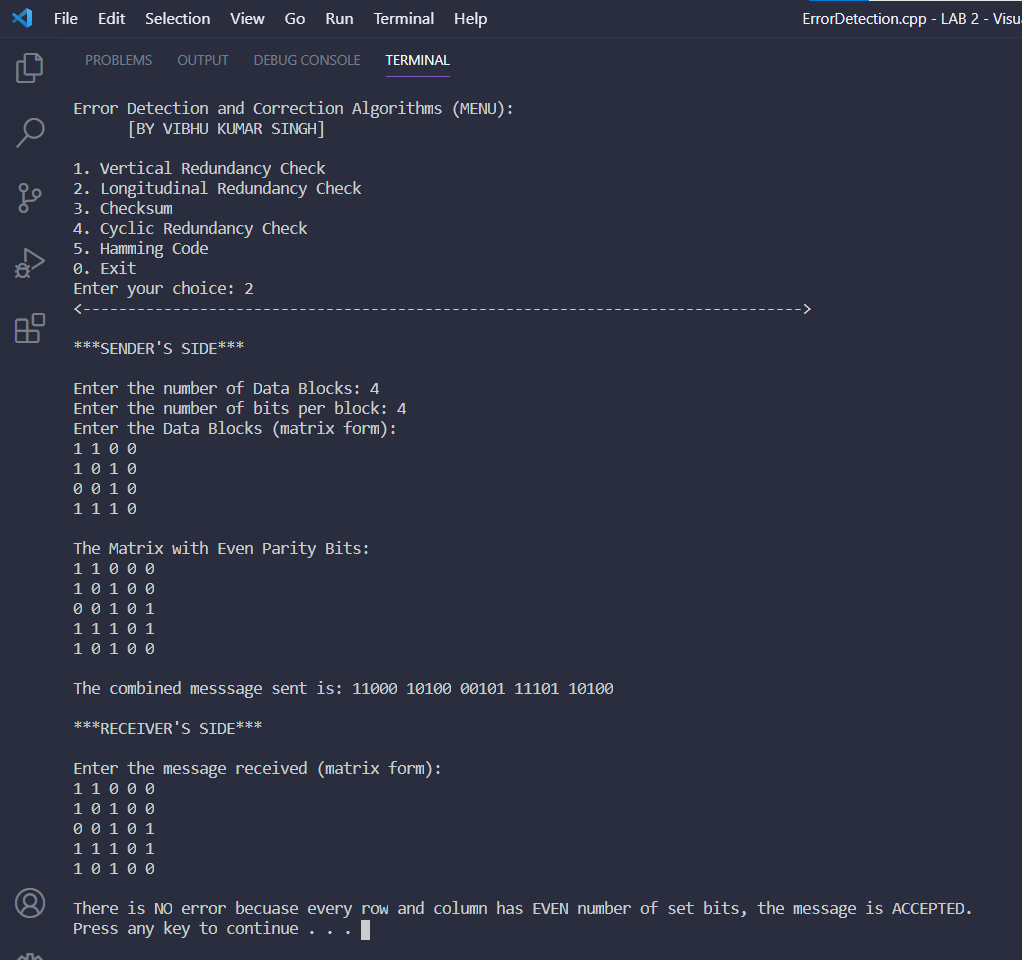


**Error Case:**

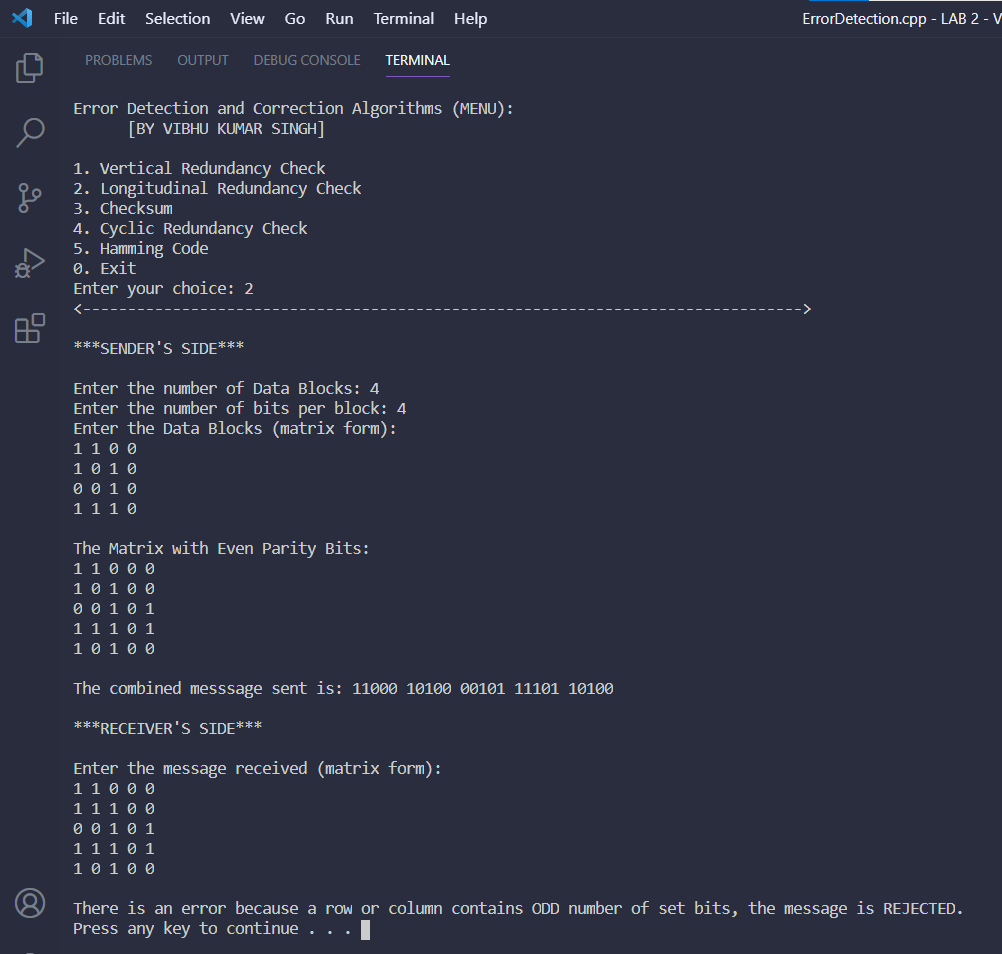
****

1. **Longitudinal Redundancy Check (LRC):**

**No-Error Case:**

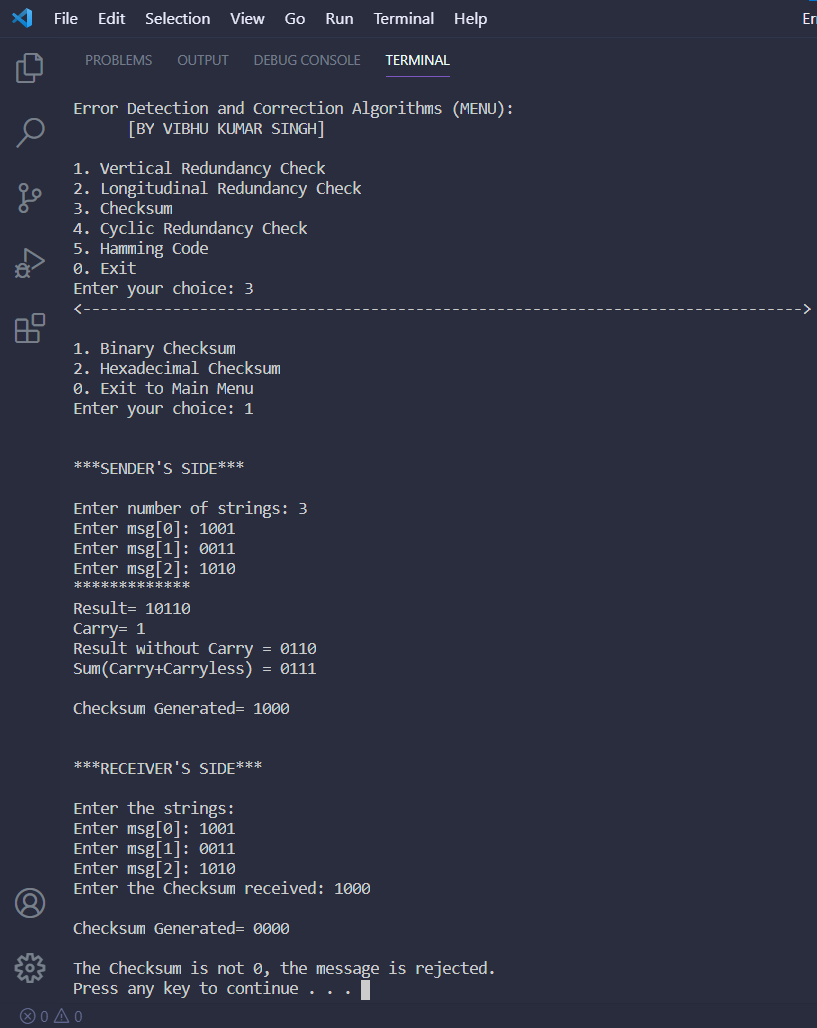
****

**Error Case:**

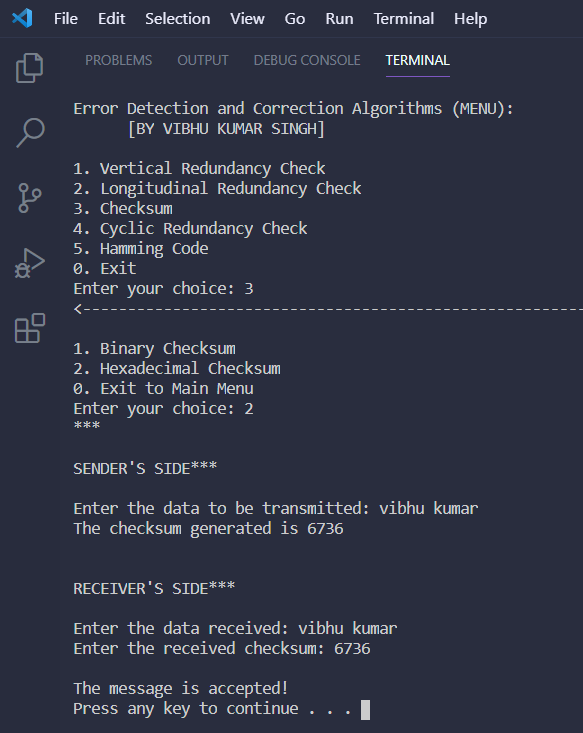
****

1. **Checksum:**

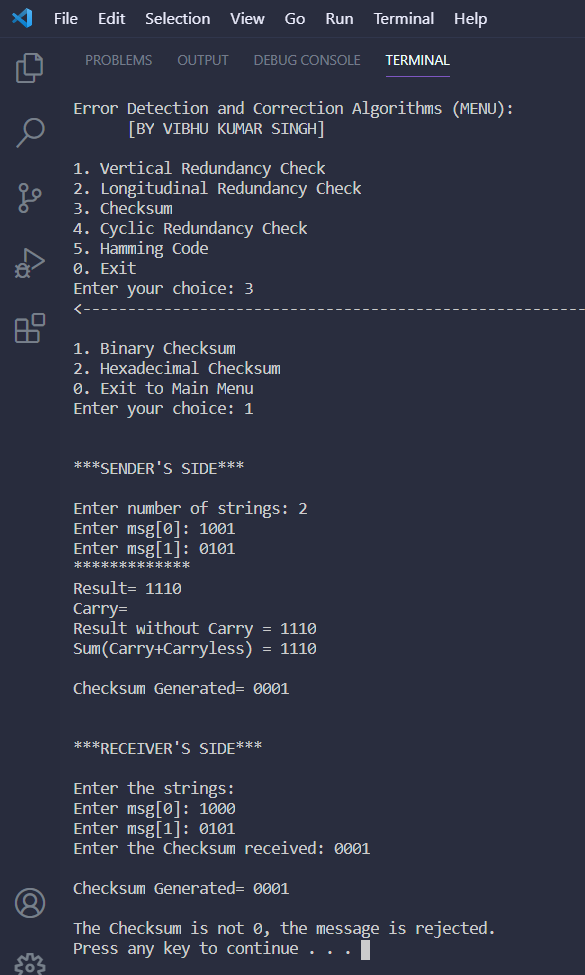
**No-Error Case(binary):**

****

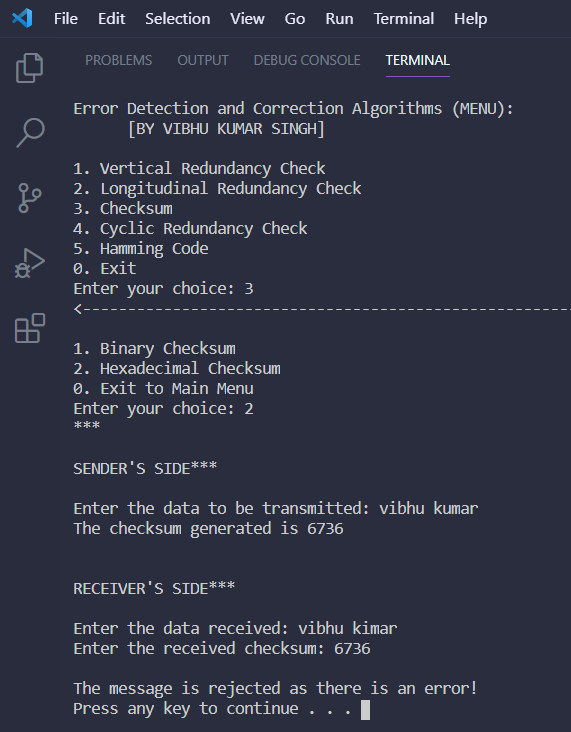
**No-Error Case(hexadecimal):**

****

**Error Case (Binary):**

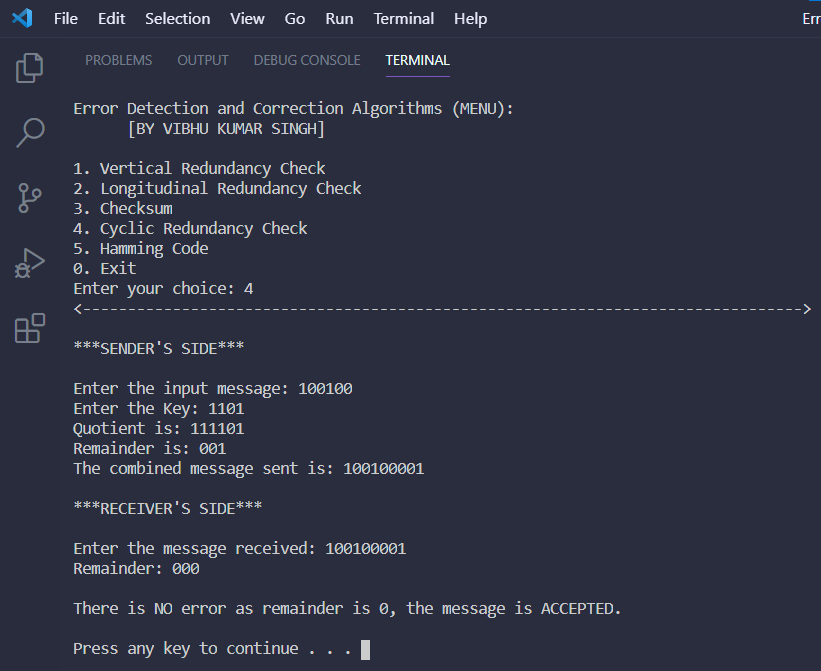
****

**Error Case(Hexadecimal):**

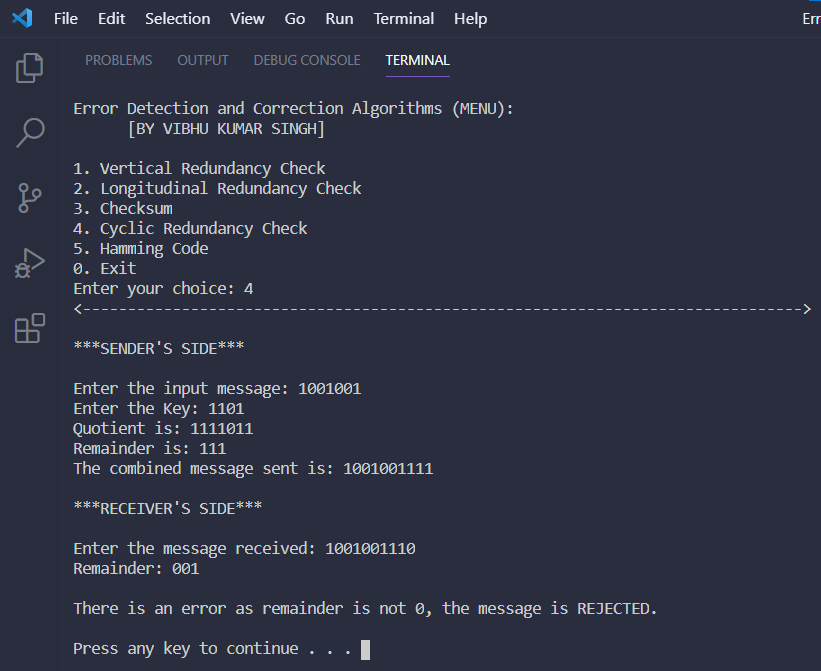
****

1. **Cyclic Redundancy Check (CRC):**

**No-Error Case:**

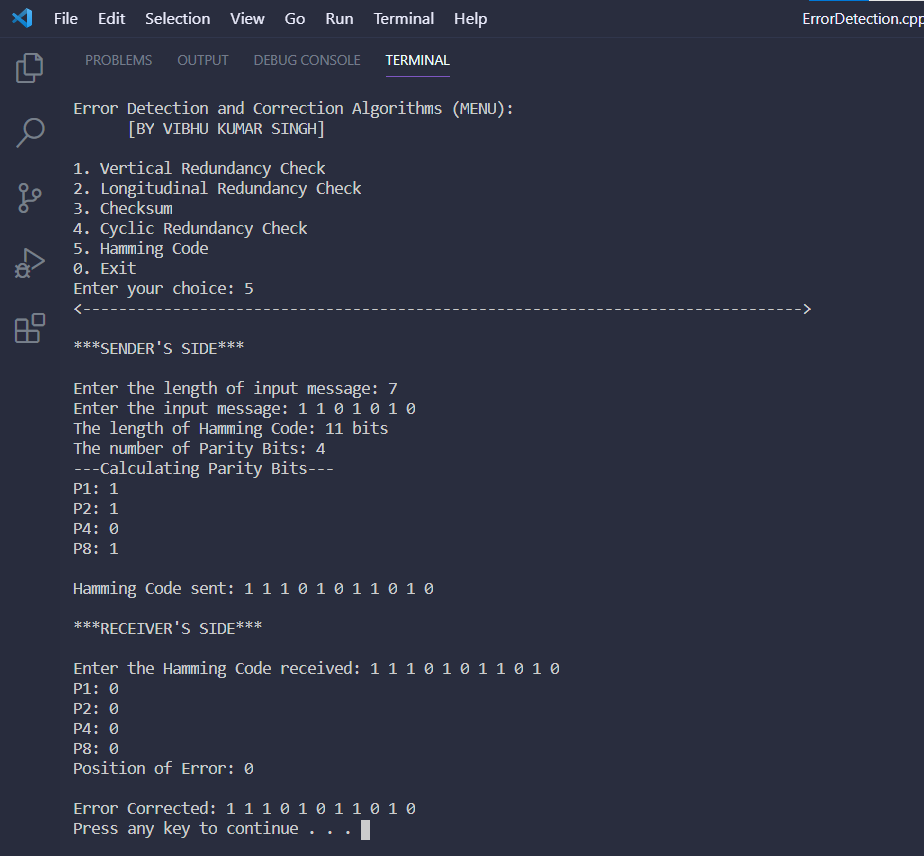
****

**Error Case:**

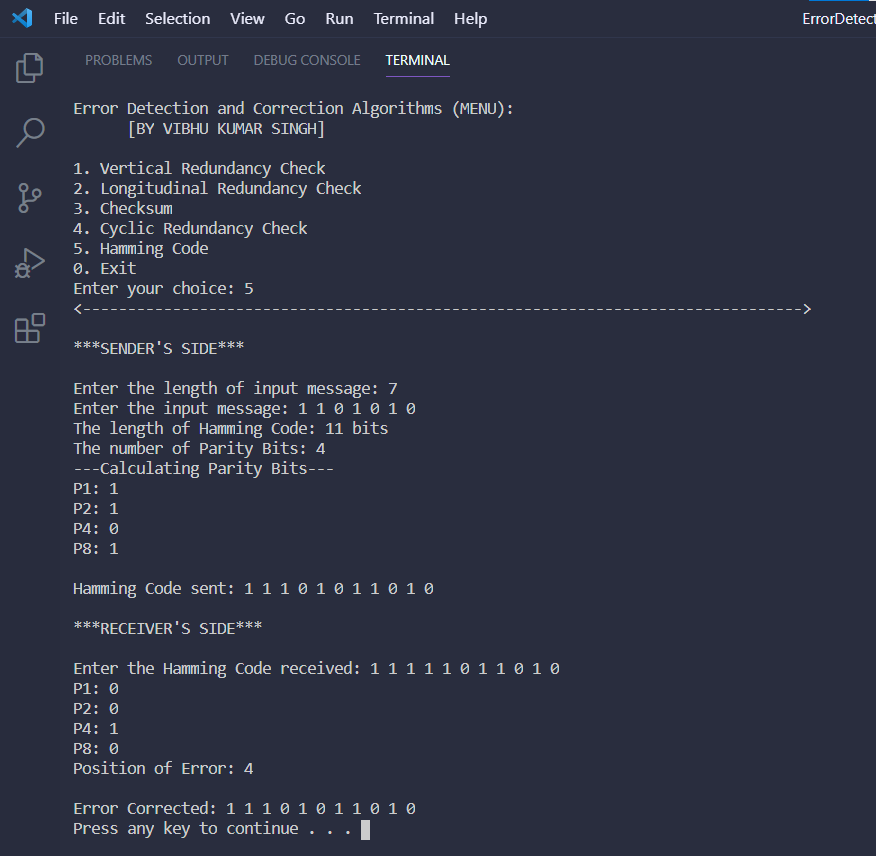
****

1. **Hamming Code:**

**No-Error Case:**

****

**Error Case:**

****

1. **Develop a menu-driven code to simulate to simulate the following flow control algorithms.**

**a) Stop and Wait**

**b) Selective Repeat**

**c) Go-back-N**

**Ans 2)**

**Aim:** To simulate Flow-Control Algorithms in Networking.

The Flow-Control Algorithms simulated are:

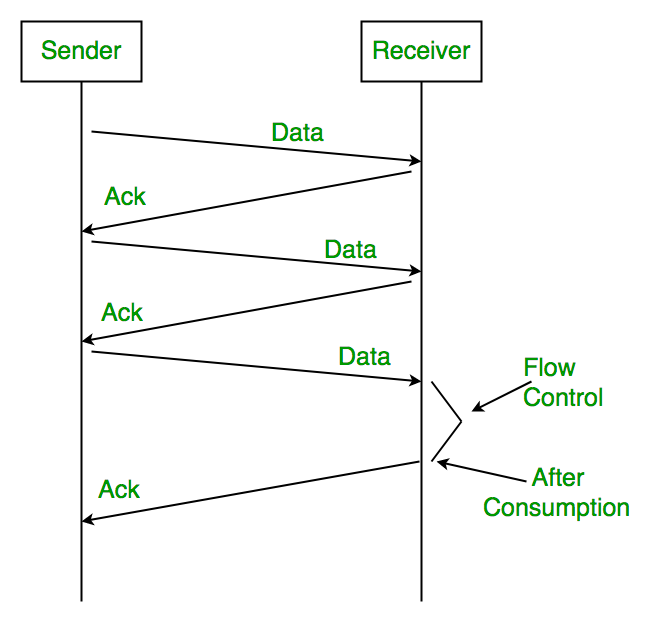
a) Stop and Wait

b) Selective Repeat

c) Go-back-N

**Algorithms:**

1. **Stop and Wait:**

****

START:

**Sender:**

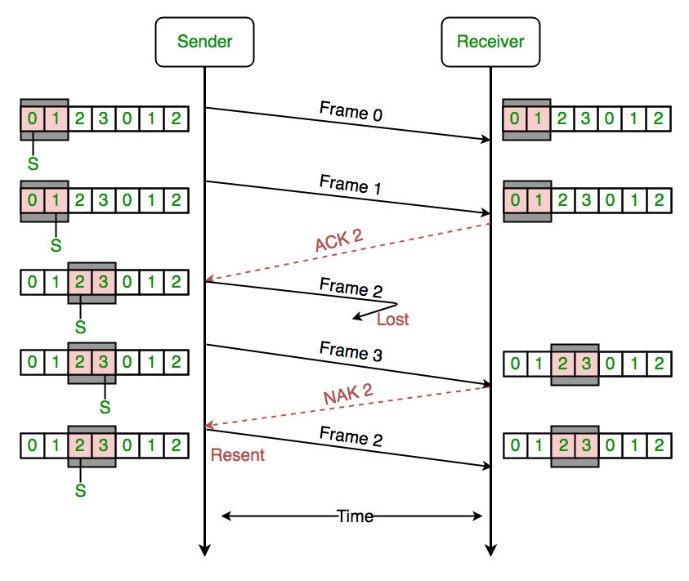
* Send one data packet at a time.
* Send next packet only after receiving acknowledgement for previous.

**Receiver:**

* Send acknowledgement after receiving and consuming of data packet.
* After consuming packet acknowledgement need to be sent (Flow Control)

END

1. **Selective Repeat:**

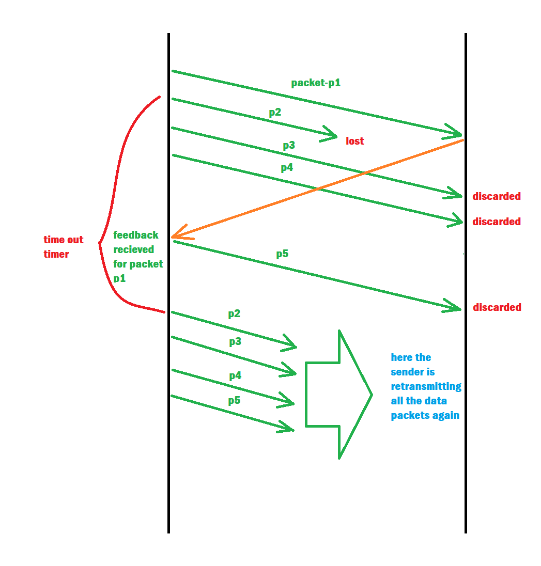
****

START:

* STEP 1: The user enters the number of frames to be sent.
* STEP 2: The size of the widow is set.
* STEP 3: Identify the number of frames to be sent at a given time.
* STEP 4: Transmit the frame.
* STEP 5: Wait and receive the acknowledgement frame.
* STEP 6: Check for the acknowledgement of each frame and repeat the process for that frame for which an acknowledgement is not received, else continue the process.
* STEP 7: Repeat the steps 4 to 6 until the number of frames to be transmitted becomes zero.

END

1. **Go-back-N:**

****

START:

* STEP 1: The user enters the number of frames to be transmitted.
* STEP 2: The size of the window is set.
* STEP 3: Identify the number of frames to be transmitted at a given time.
* STEP 4: Transmit the frames and receive the acknowledgment for the frame sent.
* STEP 5: Find the remaining frames to be sent.
* STEP 6: If for any frame, acknowledgement is not received, transmit that frame once again.
* STEP 7: Repeat the steps from 4 to 6 until the number of frames to be sent becomes zero.

END

**Menu-Driven Source Code:**

#ifdef \_WIN32

#include<windows.h>

#else

#include<unistd.h>

#endif

#include <iostream>

#include <cstdlib>

#include<bits/stdc++.h>

using namespace std;

bool flag = true;

bool flag2 = true;

int pos = 0;

int c = 0;

string ab;

void sleeping();

string arr[1000] = {"wr"};//*initialize the array*

string senderSandW(string a)

{

    if(flag)

    {

        int len = a.length();

        string b;

        b = a.substr(pos,1);

        pos++;

        flag = false;

        cout<<"Sending "<<b<<endl;

        return b;

    }

}

string receiverSandW(string a)

{

    if(flag == false)

    {

        arr[pos-1] = a;

        string c;

        if(arr[pos-1] == a){

        c = "Acknowledgement: recieved " + arr[pos-1];

        flag = true;

        return c;}

        else{

            c = "Timeout";

            senderSandW(a);

            flag = true;

            return c;

        }

    }

}

void StopWait()

{

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Stop and Wait\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";

    cout<<"Enter the String: ";

    string a;

    cin>>a;

    string temp;

    int len = a.length();

    for(int i=0;i<len;i++){

        temp = senderSandW(a);

        sleeping();

        cout<<receiverSandW(temp)<<endl;

    }

    pos = 0;

    flag = true;

    cout<<"\n\nFinished: Stop and Wait Protocol\n";

    system("pause");

}

string sendergbn(int n,string a)

{

    if(flag ==true && flag2 == true)

    {

        int len = a.length();

        string b;

        b = a.substr(pos,1);

        pos++;

        flag = false;

        cout<<"Sending "<<b<<endl;

        return b;

    }

}

void receiverbgn(int n,string a)

{

    if(flag == false)

    {

        arr[pos-1] = a;

        if(c<n)

        {

            c++;

            flag = true;

        }

        if(c>=n)

        {

            if(arr[pos-n] == "wr" )

            {

                cout<<"Error"<<endl;

                flag2 = false;

                for(int i=0;i<n;i++)

                {

                    pos = pos-n;

                    sendergbn(n,a);

                    sleeping();

                    receiverbgn(n,a);

                }

            }

            else

            {

                cout<<"\*\*\*\*\*\*\*\*\*\*"<<endl;

                cout<<"Acknowledgement: recieved "<<arr[pos-n]<<endl;

                cout<<"\*\*\*\*\*\*\*\*\*\*"<<endl;

            }

            flag = true;

        }

    }

}

void sleeping()

{

    Sleep(500);

    cout<<"sending";

    Sleep(500);

    cout<<".";

    Sleep(500);

    cout<<".";

    Sleep(500);

    cout<<"."<<endl;

}

string b[1000] = {"-"};

void SelectiveRepeat()

{

    cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SELECTIVE REPEAT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    cout<<"Enter the size of Sliding-Window: ";

    int n;

    cin>>n;

    if(n>0)

    {

        cout<<"Enter String: ";

        cin>>ab;

        string temp;

        int len = ab.length();

        for(int i=0;i<len;i++){

            temp = sendergbn(n,ab);

            sleeping();

            receiverbgn(n,temp);

        }

        for(int i=1;i<n;i++)

        {

            cout<<"Acknowledgement: recieved "<<b[i+len-n]<<endl;

        }

        pos = 0;

        flag = true;

        b[1000] = {"-"};

        c = 0;

        flag2 = true;

    }

    else

    {

        cout<<"\nEnter a valid sliding window value."<<endl;

    }

    cout<<"\n\nFinished: Selective Repeat Protocol\n";

    system("pause");

}

void gbn()

{

    cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GO BACK N\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    cout<<"Enter the size of the Sliding-Window: ";

    int n;

    cin>>n;

    if(n>0)

    {

        cout<<"Enter String: ";

        cin>>ab;

        string temp;

        int len = ab.length();

        for(int i=0;i<len;i++)

        {

            temp = sendergbn(n,ab);

            sleeping();

            receiverbgn(n,temp);

        }

        for(int i=1;i<n;i++)

        {

            cout<<"\*\*\*\*\*\*\*\*\*\*"<<endl;

            cout<<"Acknowledgement: recieved "<<arr[i+len-n]<<endl;

            cout<<"\*\*\*\*\*\*\*\*\*\*"<<endl;

        }

        pos = 0;

        flag = true;

        arr[1000] = {"wr"};

        c = 0;

        flag2 = true;

    }

    else

    {

        cout<<"\nEnter a valid sliding window value."<<endl;

    }

    cout<<"\n\nFinished: Go-Back-N  Protocol\n";

    system("pause");

}

int main()

{

    while(1)

    {

        start:

        system("cls");

        int choice;

        cout<<"Flow Conrol Protocols (MENU): \n [BY VIBHU KUMAR SINGH]\n\n1. Stop and Wait\n2. Selective Repeat\n3. Go-Back-N\n0. Exit\nEnter your choice: ";

        cin>>choice;

        cout<<"<-------------------------------------------------------------------------------->\n\n";

        switch(choice)

        {

            case 1:

                StopWait();

            break;

            case 2:

                SelectiveRepeat();

            break;

            case 3:

            start\_checksum:

                gbn();

            break;

            case 0:

                exit(0);

            break;

            default:

                cout<<"Invalid input";

                goto start;

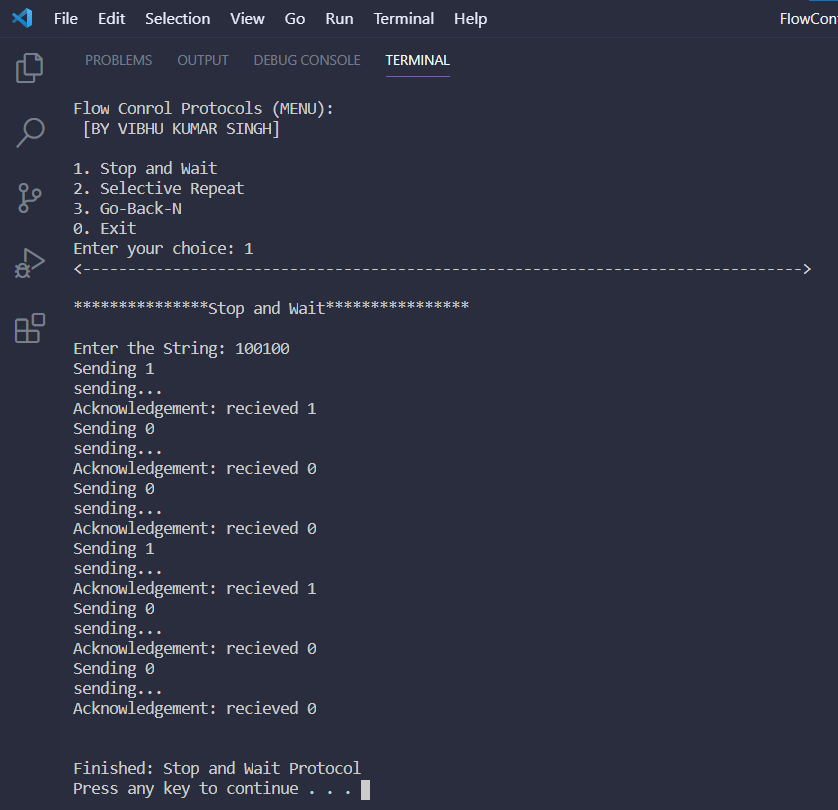
        }

    }

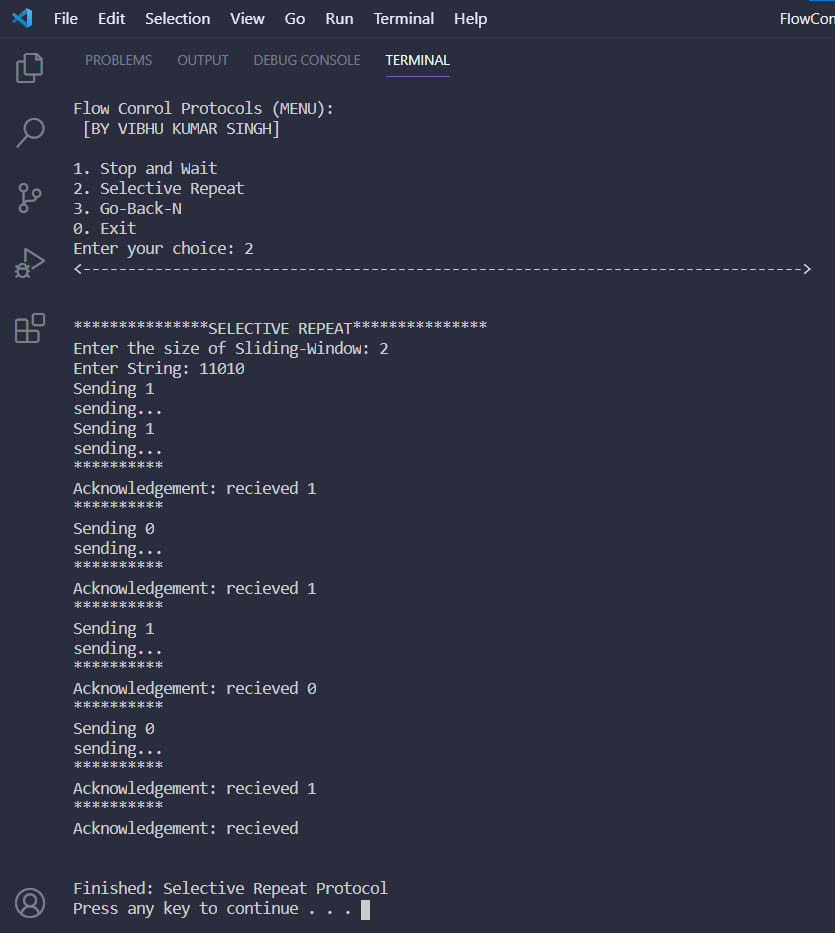
}

**OUTPUT SCREENSHOTS:**

1. **Stop and Wait:**

****

1. **Selective Repeat:**

****

1. **Go-Back-N:**

