



# VIT<sup>®</sup>

## Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

### 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)**

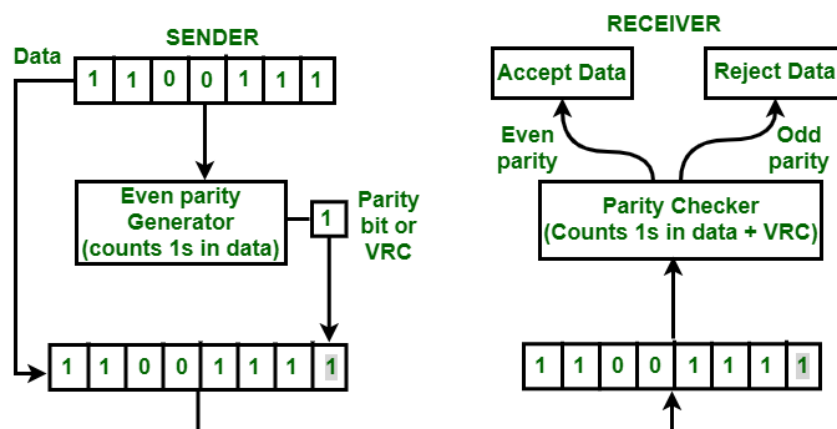
**Aim:** To simulate Error Detection algorithms in Networking.

The Detection Algorithms simulated are:

- a) VRC
- b) LRC
- c) Checksum
- d) CRC
- e) Hamming Code

**Algorithms:**

## a) Vertical Redundancy Check (VRC):

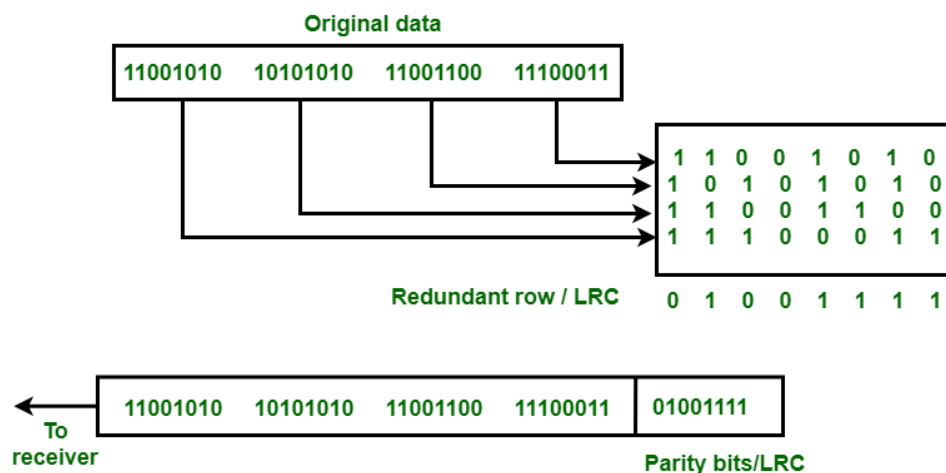


**START:**

- 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

### b) Longitudinal Redundancy Check (LRC):

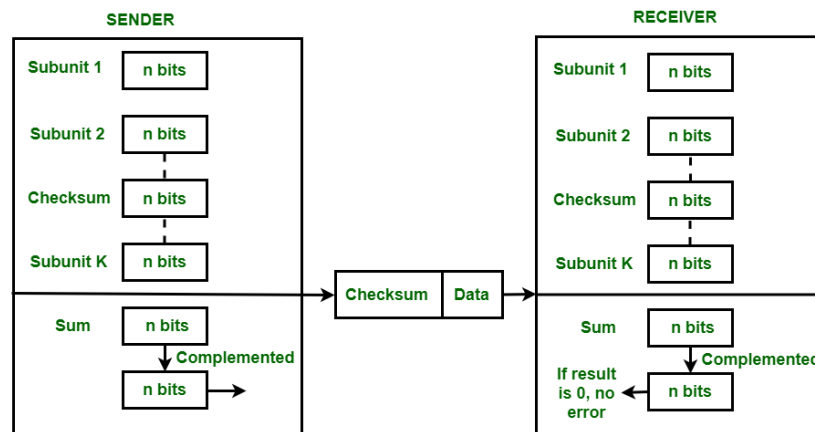


START:

- 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

### c) Checksum:

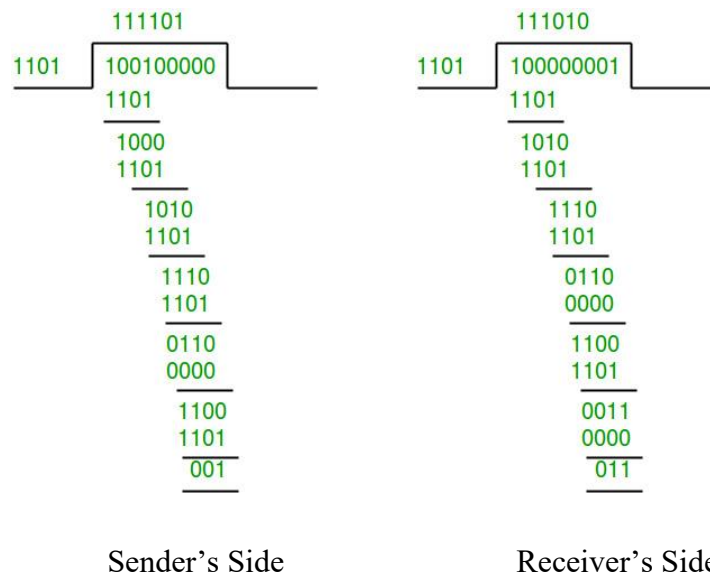


START:

- 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

#### d) Cyclic Redundancy Check (CRC):

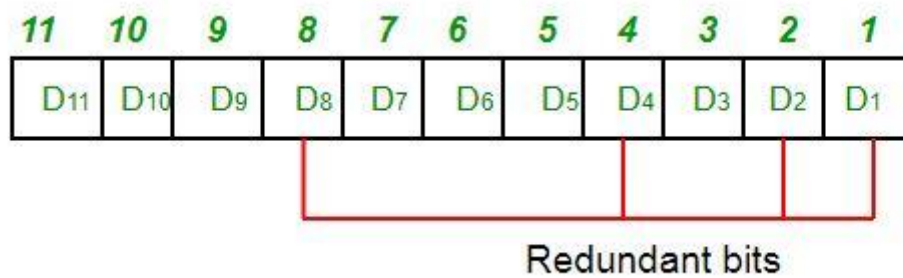


START:

- 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

### e) 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<<"\n\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<<"\n\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<<"\n\nThere is an error because a row or column contains ODD number of set bits
, the message is REJECTED.\n";
}
else
{
    cout<<"\n\nThere is NO error becuae every row and column has EVEN number of set bit
s, 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 S
INGH]\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:

```

```

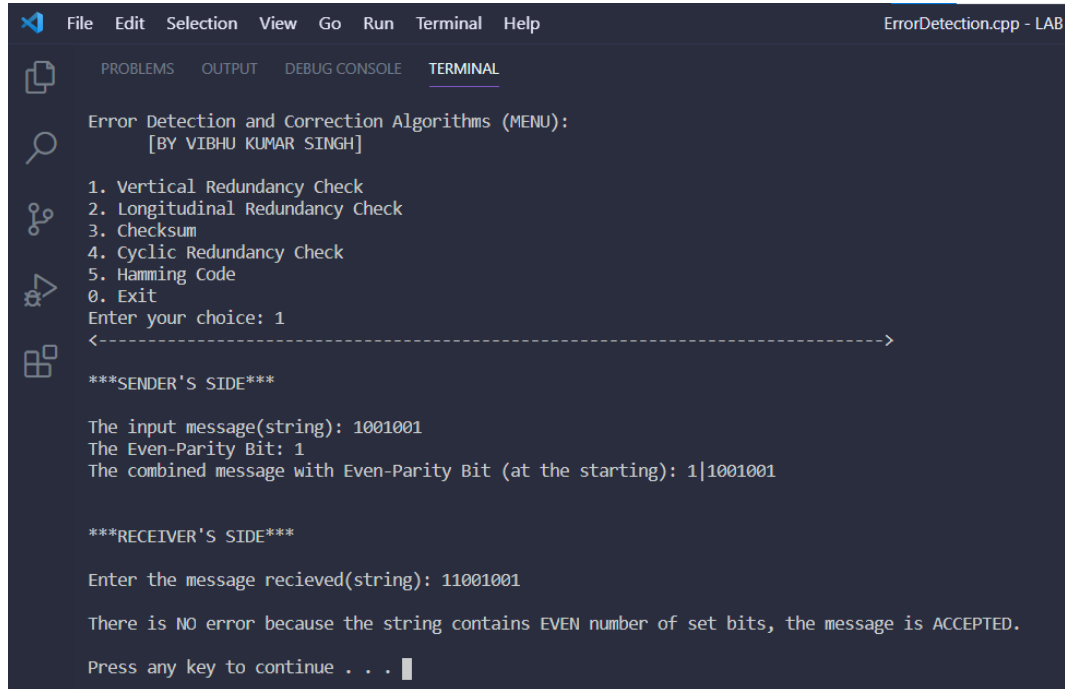
    cout<<"Invalid input";
    goto start;
}
}
}

```

## OUTPUT SCREENSHOTS:

### a) Vertical Redundancy Check (VRC):

#### No-Error Case:

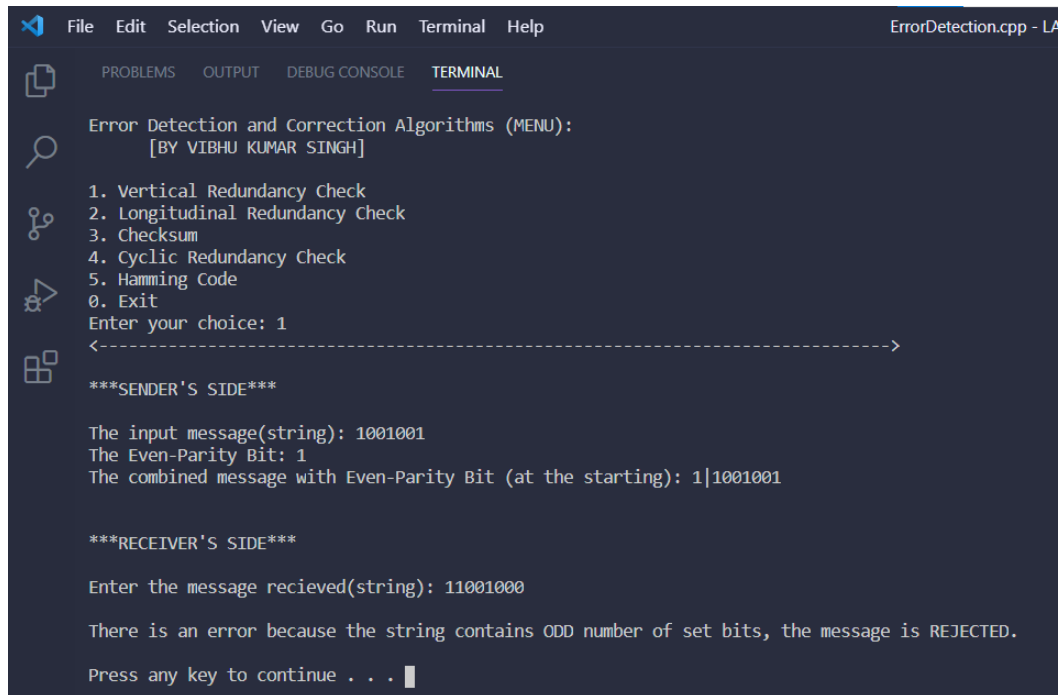


```

ErrorDetection.cpp - LAB
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]
1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 1
<----->
***SENDER'S SIDE***
The input message(string): 1001001
The Even-Parity Bit: 1
The combined message with Even-Parity Bit (at the starting): 1|1001001
***RECEIVER'S SIDE***
Enter the message recieved(string): 11001001
There is NO error because the string contains EVEN number of set bits, the message is ACCEPTED.
Press any key to continue . . .

```

#### Error Case:



```

ErrorDetection.cpp - LAB
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]
1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 1
<----->
***SENDER'S SIDE***
The input message(string): 1001001
The Even-Parity Bit: 1
The combined message with Even-Parity Bit (at the starting): 1|1001001
***RECEIVER'S SIDE***
Enter the message recieved(string): 11001000
There is an error because the string contains ODD number of set bits, the message is REJECTED.
Press any key to continue . . .

```

## b) Longitudinal Redundancy Check (LRC):

### No-Error Case:

```
File Edit Selection View Go Run Terminal Help ErrorDetection.cpp - LAB 2 - Visu
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]
1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 2
<----->

***SENDER'S SIDE***

Enter the number of Data Blocks: 4
Enter the number of bits per block: 4
Enter the Data Blocks (matrix form):
1 1 0 0
1 0 1 0
0 0 1 0
1 1 1 0

The Matrix with Even Parity Bits:
1 1 0 0 0
1 0 1 0 0
0 0 1 0 1
1 1 1 0 1
1 0 1 0 0

The combined message sent is: 11000 10100 00101 11101 10100

***RECEIVER'S SIDE***

Enter the message received (matrix form):
1 1 0 0 0
1 0 1 0 0
0 0 1 0 1
1 1 1 0 1
1 0 1 0 0

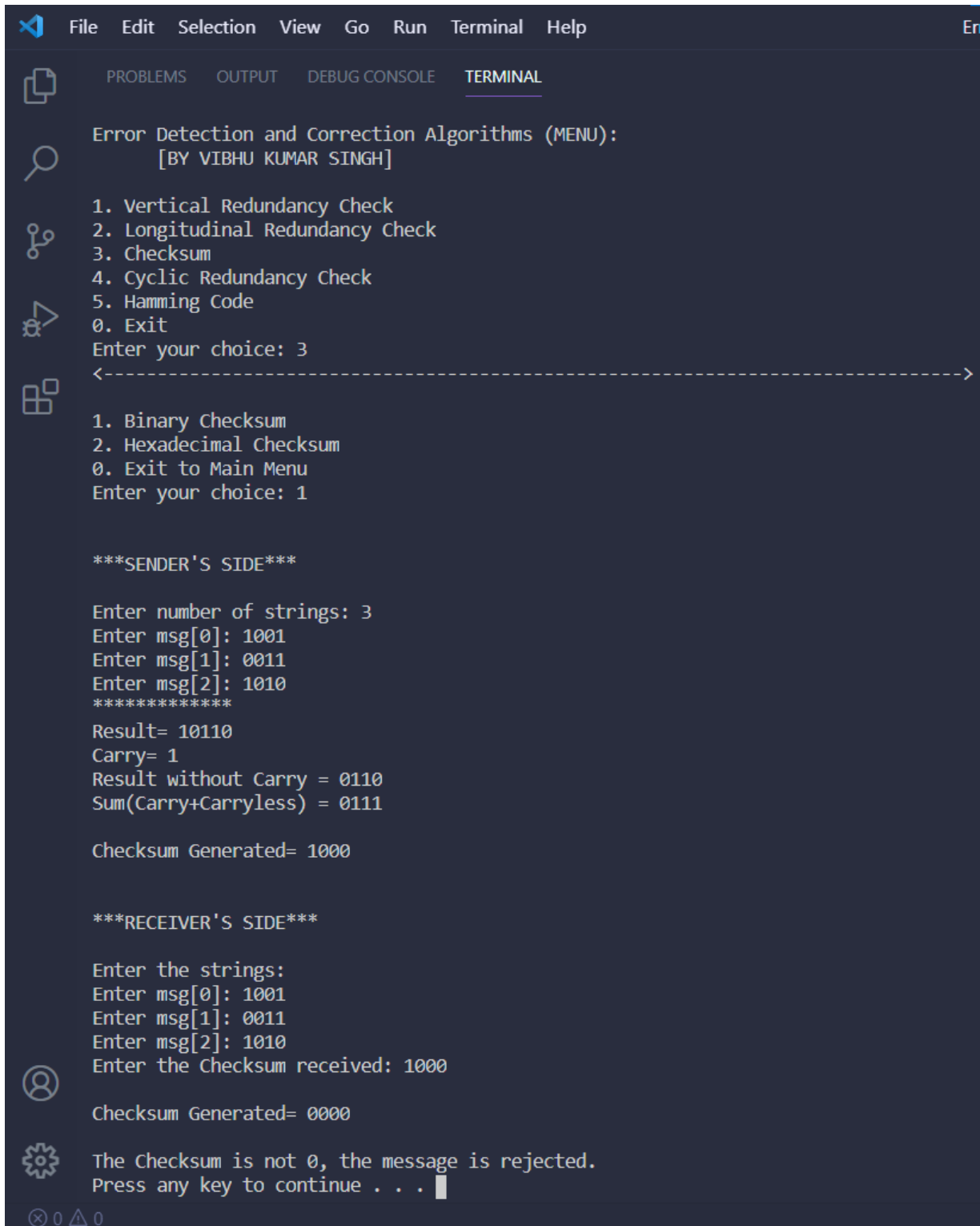
There is NO error becuae every row and column has EVEN number of set bits, the message is ACCEPTED.
Press any key to continue . . .
```

## Error Case:

```
File Edit Selection View Go Run Terminal Help ErrorDetection.cpp - LAB 2 - V
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]
1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 2
<----->
***SENDER'S SIDE***
Enter the number of Data Blocks: 4
Enter the number of bits per block: 4
Enter the Data Blocks (matrix form):
1 1 0 0
1 0 1 0
0 0 1 0
1 1 1 0
The Matrix with Even Parity Bits:
1 1 0 0 0
1 0 1 0 0
0 0 1 0 1
1 1 1 0 1
1 0 1 0 0
The combined message sent is: 11000 10100 00101 11101 10100
***RECEIVER'S SIDE***
Enter the message received (matrix form):
1 1 0 0 0
1 1 1 0 0
0 0 1 0 1
1 1 1 0 1
1 0 1 0 0
There is an error because a row or column contains ODD number of set bits, the message is REJECTED.
Press any key to continue . . .
```

### c) Checksum:

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



```
File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]

1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 3
<----->

1. Binary Checksum
2. Hexadecimal Checksum
0. Exit to Main Menu
Enter your choice: 1

***SENDER'S SIDE***

Enter number of strings: 3
Enter msg[0]: 1001
Enter msg[1]: 0011
Enter msg[2]: 1010
*****
Result= 10110
Carry= 1
Result without Carry = 0110
Sum(Carry+Carryless) = 0111

Checksum Generated= 1000

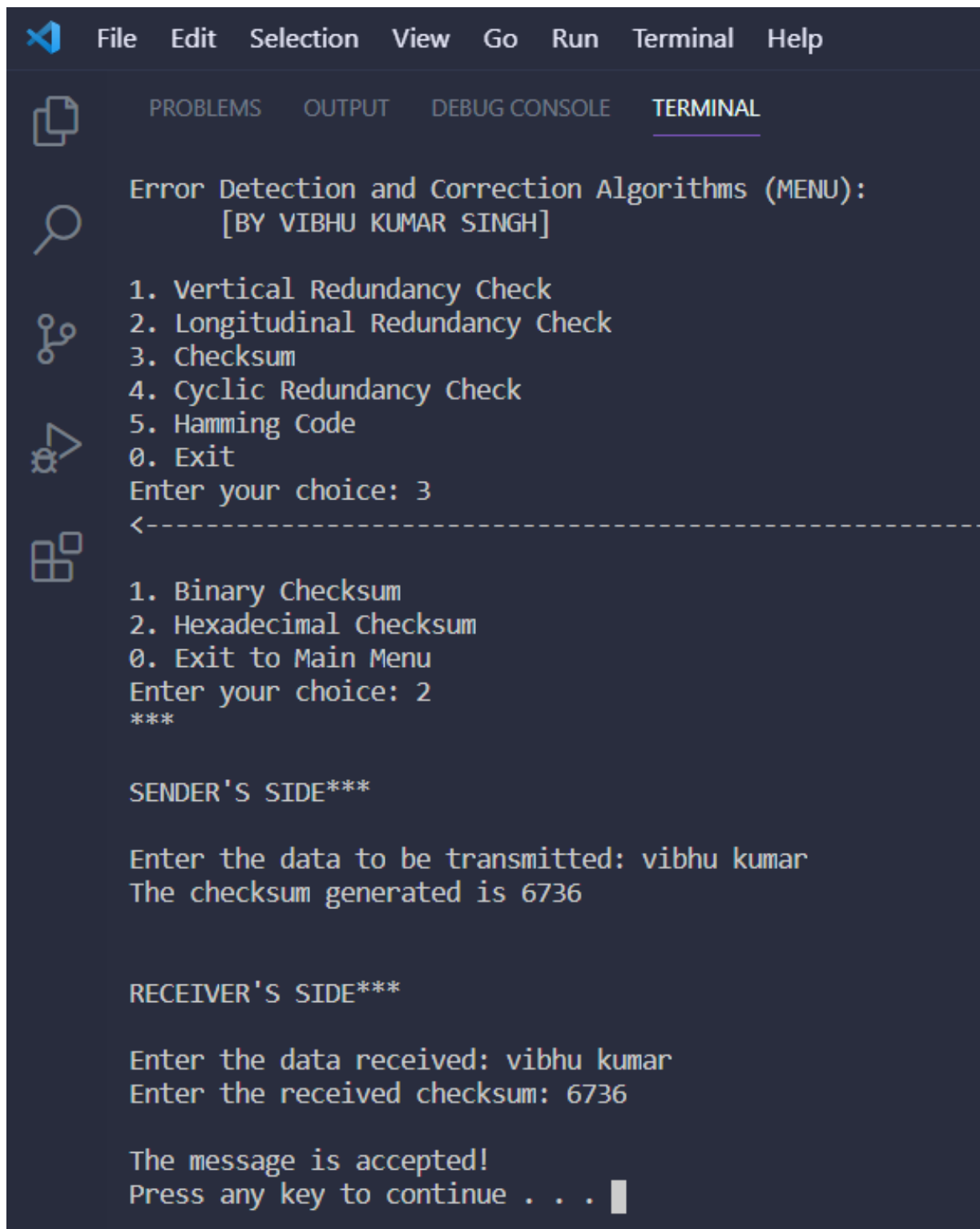
***RECEIVER'S SIDE***

Enter the strings:
Enter msg[0]: 1001
Enter msg[1]: 0011
Enter msg[2]: 1010
Enter the Checksum received: 1000

Checksum Generated= 0000

The Checksum is not 0, the message is rejected.
Press any key to continue . . .
```

## No-Error Case(hexadecimal):



```
File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]

1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 3
<-----

1. Binary Checksum
2. Hexadecimal Checksum
0. Exit to Main Menu
Enter your choice: 2
***

SENDER'S SIDE***

Enter the data to be transmitted: vibhu kumar
The checksum generated is 6736

RECEIVER'S SIDE***

Enter the data received: vibhu kumar
Enter the received checksum: 6736

The message is accepted!
Press any key to continue . . .
```

## Error Case (Binary):

```
File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]

1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 3
<-----

1. Binary Checksum
2. Hexadecimal Checksum
0. Exit to Main Menu
Enter your choice: 1

***SENDER'S SIDE***

Enter number of strings: 2
Enter msg[0]: 1001
Enter msg[1]: 0101
*****
Result= 1110
Carry=
Result without Carry = 1110
Sum(Carry+Carryless) = 1110

Checksum Generated= 0001

***RECEIVER'S SIDE***

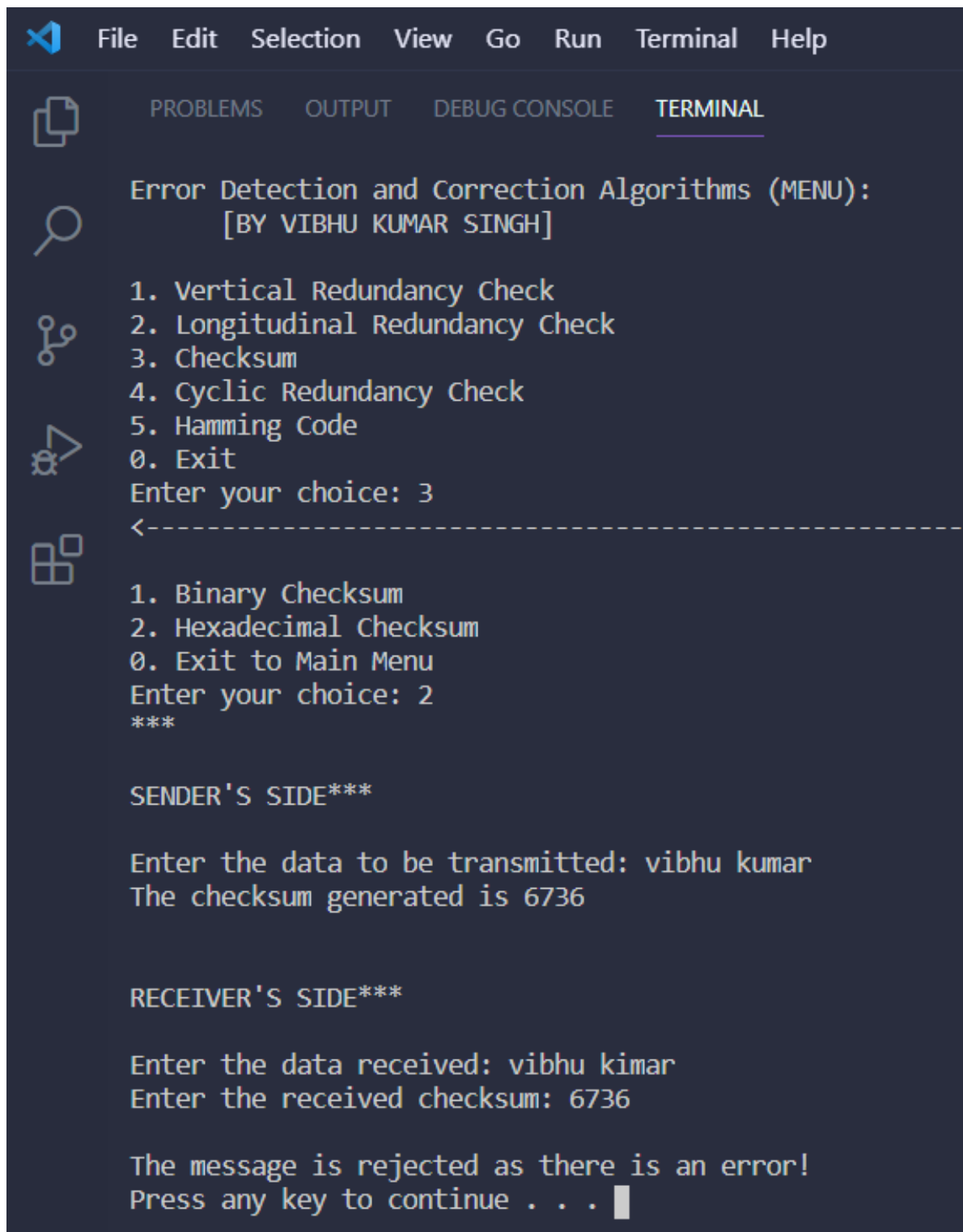
Enter the strings:
Enter msg[0]: 1000
Enter msg[1]: 0101
Enter the Checksum received: 0001

Checksum Generated= 0001

The Checksum is not 0, the message is rejected.
Press any key to continue . . .
```



## Error Case(Hexadecimal):



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]

1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 3
<-----

1. Binary Checksum
2. Hexadecimal Checksum
0. Exit to Main Menu
Enter your choice: 2
***

SENDER'S SIDE***

Enter the data to be transmitted: vibhu kumar
The checksum generated is 6736

RECEIVER'S SIDE***

Enter the data received: vibhu kimar
Enter the received checksum: 6736

The message is rejected as there is an error!
Press any key to continue . . .
```

## d) Cyclic Redundancy Check (CRC):

### No-Error Case:

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]

1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 4
<----->

***SENDER'S SIDE***

Enter the input message: 100100
Enter the Key: 1101
Quotient is: 111101
Remainder is: 001
The combined message sent is: 100100001

***RECEIVER'S SIDE***

Enter the message received: 100100001
Remainder: 000

There is NO error as remainder is 0, the message is ACCEPTED.

Press any key to continue . . .
```

### Error Case:

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]

1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 4
<----->

***SENDER'S SIDE***

Enter the input message: 1001001
Enter the Key: 1101
Quotient is: 1111011
Remainder is: 111
The combined message sent is: 1001001111

***RECEIVER'S SIDE***

Enter the message received: 1001001110
Remainder: 001

There is an error as remainder is not 0, the message is REJECTED.

Press any key to continue . . .
```

## e) Hamming Code: No-Error Case:

```
File Edit Selection View Go Run Terminal Help ErrorDetection.cpp
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]
1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 5
<----->

***SENDER'S SIDE***

Enter the length of input message: 7
Enter the input message: 1 1 0 1 0 1 0
The length of Hamming Code: 11 bits
The number of Parity Bits: 4
---Calculating Parity Bits---
P1: 1
P2: 1
P4: 0
P8: 1

Hamming Code sent: 1 1 1 0 1 0 1 1 0 1 0

***RECEIVER'S SIDE***

Enter the Hamming Code received: 1 1 1 0 1 0 1 1 0 1 0
P1: 0
P2: 0
P4: 0
P8: 0
Position of Error: 0

Error Corrected: 1 1 1 0 1 0 1 1 0 1 0
Press any key to continue . . .
```

## Error Case:

```
File Edit Selection View Go Run Terminal Help ErrorDetect
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Error Detection and Correction Algorithms (MENU):
[BY VIBHU KUMAR SINGH]
1. Vertical Redundancy Check
2. Longitudinal Redundancy Check
3. Checksum
4. Cyclic Redundancy Check
5. Hamming Code
0. Exit
Enter your choice: 5
<----->

***SENDER'S SIDE***

Enter the length of input message: 7
Enter the input message: 1 1 0 1 0 1 0
The length of Hamming Code: 11 bits
The number of Parity Bits: 4
---Calculating Parity Bits---
P1: 1
P2: 1
P4: 0
P8: 1

Hamming Code sent: 1 1 1 0 1 0 1 1 0 1 0

***RECEIVER'S SIDE***

Enter the Hamming Code received: 1 1 1 1 0 1 1 0 1 0
P1: 0
P2: 0
P4: 1
P8: 0
Position of Error: 4

Error Corrected: 1 1 1 0 1 0 1 1 0 1 0
Press any key to continue . . .
```

## 2. Develop a menu-driven code to simulate the following flow control algorithms.

- a) Stop and Wait
- b) Selective Repeat
- c) Go-back-N

**Ans 2)**

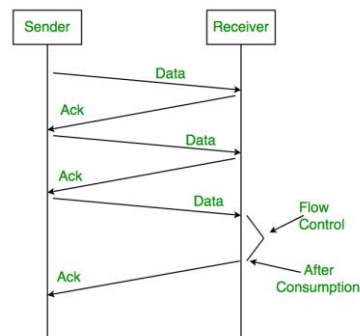
**Aim:** To simulate Flow-Control Algorithms in Networking.

The Flow-Control Algorithms simulated are:

- a) Stop and Wait
- b) Selective Repeat
- c) Go-back-N

**Algorithms:**

**a) Stop and Wait:**



**START:**

**Sender:**

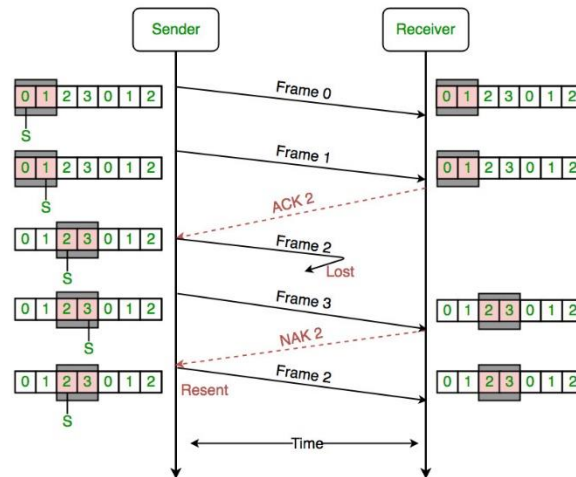
- Send one data packet at a time.
- Send next packet only after receiving acknowledgement for previous.

**Receiver:**

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

**END**

## b) Selective Repeat:

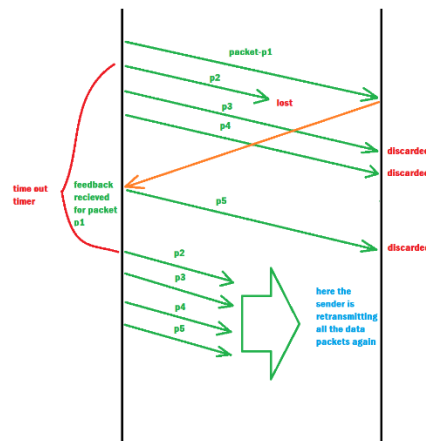


START:

- STEP 1: The user enters the number of frames to be sent.
- STEP 2: The size of the window 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

### c) 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;

```

```

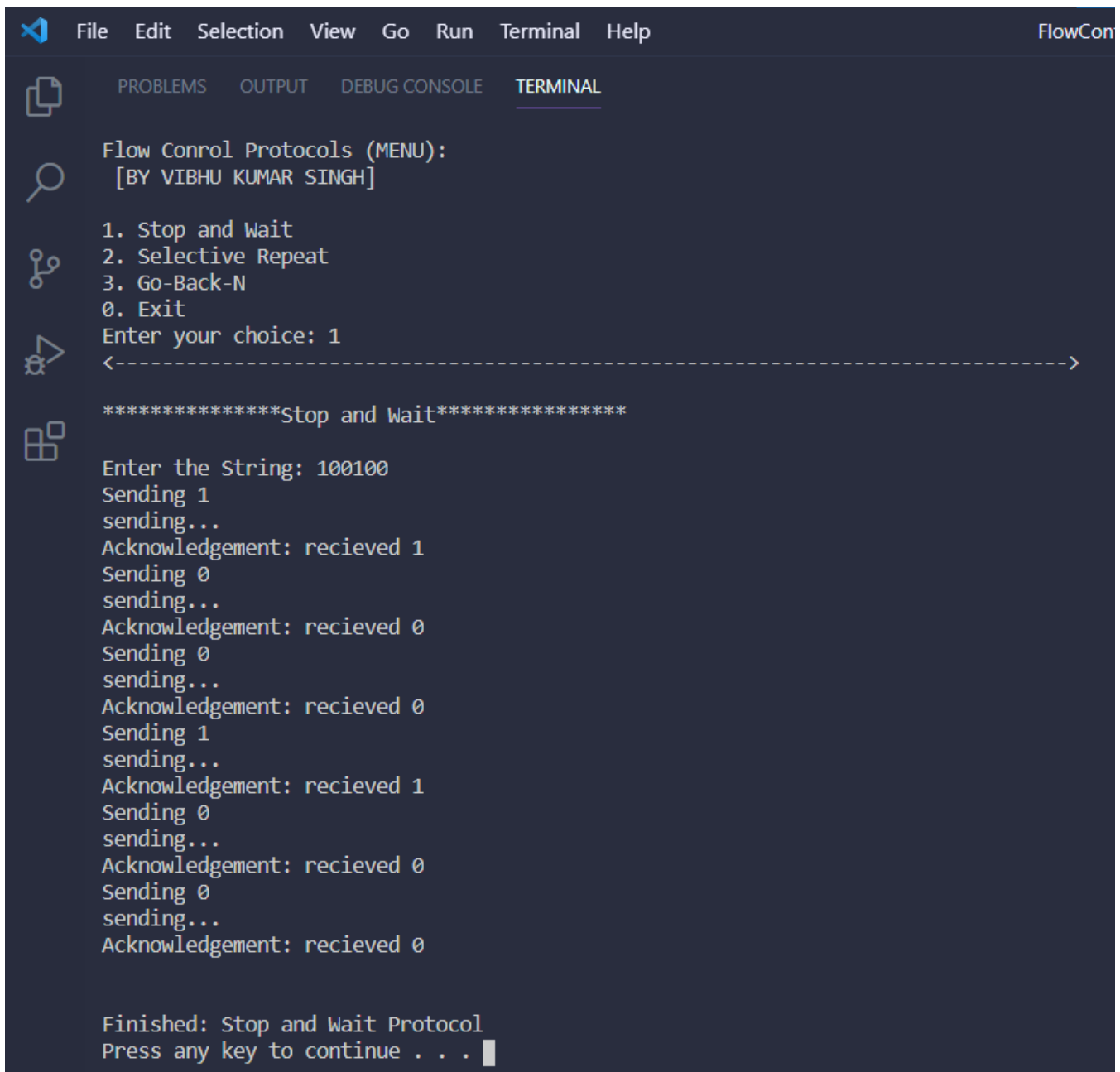
        cout<<"Acknowledgement: recieved "<<arr[i+len-n]<<endl;
        cout<<"*****"<<endl;
    }
    pos = 0;
    flag = true;
    arr[1000] = {"wr"};
    c = 0;
    flag2 = true;
}
else
{
    cout<<"\nEnter a valid sliding window value."<<endl;
}
cout<<"\n\nFinished: Go-Back-N Protocol\n";
system("pause");
}

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:

### a) Stop and Wait:



```
File Edit Selection View Go Run Terminal Help FlowCon
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Flow Control Protocols (MENU):
[BY VIBHU KUMAR SINGH]
1. Stop and Wait
2. Selective Repeat
3. Go-Back-N
0. Exit
Enter your choice: 1
<----->

*****Stop and Wait*****
Enter the String: 100100
Sending 1
sending...
Acknowledgement: recieved 1
Sending 0
sending...
Acknowledgement: recieved 0
Sending 0
sending...
Acknowledgement: recieved 0
Sending 1
sending...
Acknowledgement: recieved 1
Sending 0
sending...
Acknowledgement: recieved 0
Sending 0
sending...
Acknowledgement: recieved 0

Finished: Stop and Wait Protocol
Press any key to continue . . .
```

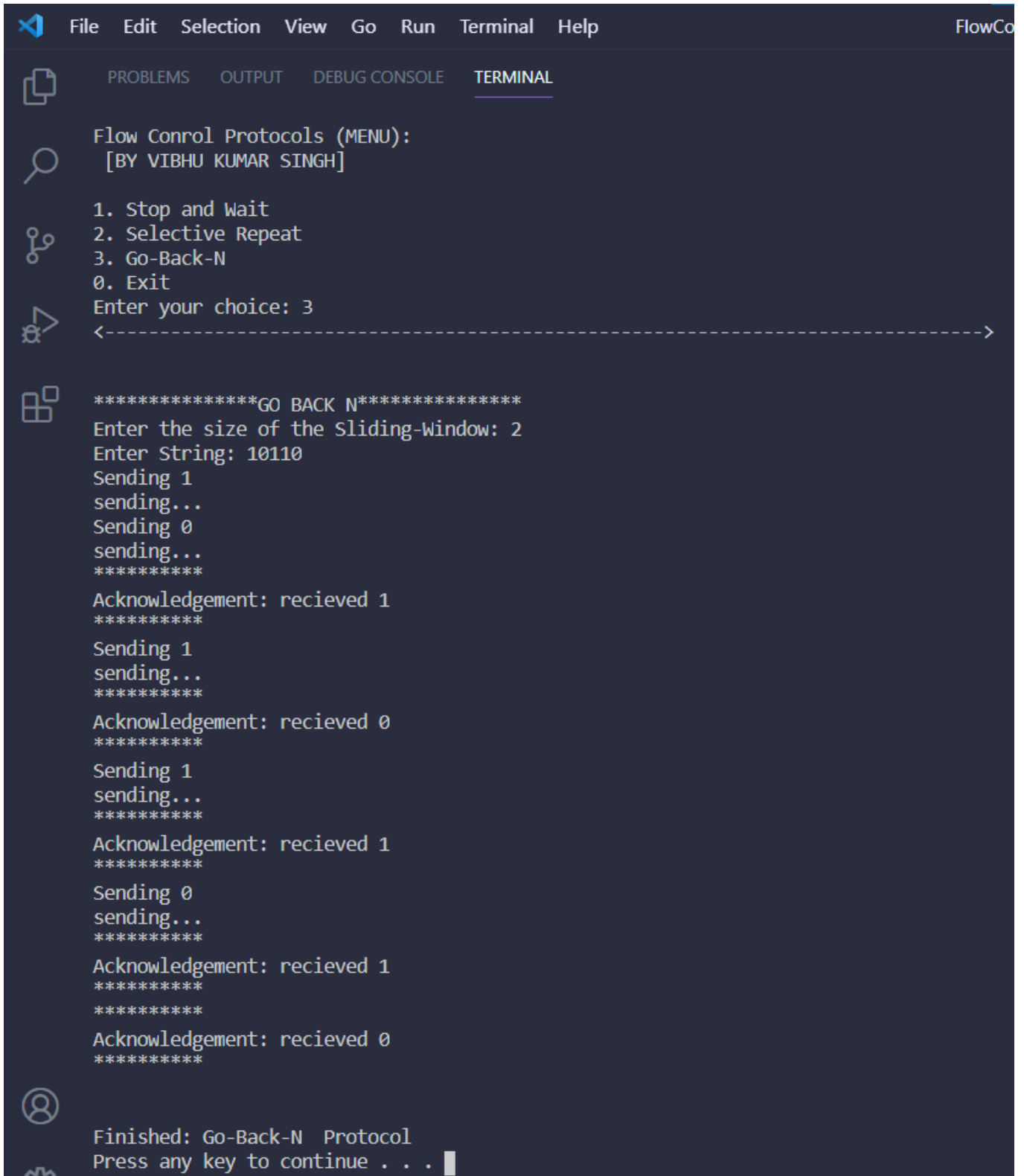
## b) Selective Repeat:

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Flow Control Protocols (MENU):
[BY VIBHU KUMAR SINGH]
1. Stop and Wait
2. Selective Repeat
3. Go-Back-N
0. Exit
Enter your choice: 2
<----->

*****SELECTIVE REPEAT*****
Enter the size of Sliding-Window: 2
Enter String: 11010
Sending 1
sending...
Sending 1
sending...
*****
Acknowledgement: recieved 1
*****
Sending 0
sending...
*****
Acknowledgement: recieved 1
*****
Sending 1
sending...
*****
Acknowledgement: recieved 0
*****
Sending 0
sending...
*****
Acknowledgement: recieved 1
*****
Acknowledgement: recieved

Finished: Selective Repeat Protocol
Press any key to continue . . .
```

### c) Go-Back-N:



```
Flow Control Protocols (MENU):  
[BY VIBHU KUMAR SINGH]  
  
1. Stop and Wait  
2. Selective Repeat  
3. Go-Back-N  
0. Exit  
Enter your choice: 3  
----->  
  
*****GO BACK N*****  
Enter the size of the Sliding-Window: 2  
Enter String: 10110  
Sending 1  
sending...  
Sending 0  
sending...  
*****  
Acknowledgement: recieved 1  
*****  
Sending 1  
sending...  
*****  
Acknowledgement: recieved 0  
*****  
Sending 1  
sending...  
*****  
Acknowledgement: recieved 1  
*****  
Sending 0  
sending...  
*****  
Acknowledgement: recieved 1  
*****  
*****  
Acknowledgement: recieved 0  
*****  
  
Finished: Go-Back-N Protocol  
Press any key to continue . . .
```