**NAME : AKHILA GURUJU**

**ENROLL : 2020BITE062**

**DATA COMMUNICATION ASSIGNMENT.**

**CODE :**

/\*

Digital data generator: generates completely random data sequence and

a random sequence with some fixed sub-sequences like eight consecutive

zeros.

\*/

#include<bits/stdc++.h>

using namespace std;

string lcs(string &x,string &y)

{

int m=x.size();

int n=y.size();

int dp[m+1][n+1];

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

{

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

{

if(i==0 || j==0)

dp[i][j]=0;

else if(x[i-1]==y[j-1])

dp[i][j]=dp[i-1][j-1]+1;

else

dp[i][j]=max(dp[i-1][j],dp[i][j-1]);

}

}

int index=dp[m][n];

string temp(index+1,'\0');

int i=m,j=n;

while(i>0 && j>0)

{

if(x[i-1]==y[j-1])

{

temp[index-1]=x[i-1];

i--;

j--;

index--;

}

else if(dp[i-1][j]>dp[i][j-1])

i--;

else

j--;

}

return temp;

}

string LongPalin(string &str)

{

string rev=str;

reverse(rev.begin(),rev.end());

return lcs(str,rev);

}

// longest common subsequence function

//we will find a sequence of 4 or 8 zeroes

// int LCS(int ptrn1[], int ptrn2[], int n, int m)

// {

// int idx=-1;

// int dp[n + 1][m + 1];

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

// for (int j = 0; j <= m; j++)

// dp[i][j] = 0;

// for (int i = n - 1; i >= 0; i--)

// {

// for (int j = m - 1; j >= 0; j--)

// {

// if (ptrn1[i] == ptrn2[j])

// {

// dp[i][j] = dp[i + 1][j + 1] + 1;

// if(dp[i][j]==m)

// idx=i;

// }

// }

// }

// return idx;

// }

// four consecutive zeroes function

void FourBitsZero(int\* s,int size,int\* x)

{

for(int i=0;i<10;i++)

x[i]=-1;

int srt=0,cnt=0,i=0,j=0;

while(i<size)

{

while(i<size && s[i]==1)

{

i++;

}

srt=i;

while(i<size && s[i]==0 && cnt<4)

{

cnt++;

i++;

}

if(cnt==4)

{

cnt=0;

x[j]=srt;

j++;

}

else

cnt=0;

}

}

// 8 consecutive zeroes function

void EightBitsZero(int\* s,int size,int\* x)

{

for(int i=0;i<3;i++)

x[i]=-1;

int srt=0,cnt=0,i=0,j=0;

while(i<size)

{

while(i<size && s[i]==1)

{

i++;

}

srt=i;

while(i<size && s[i]==0 && cnt<8)

{

cnt++;

i++;

}

if(cnt==8)

{

cnt=0;

x[j]=srt;

j++;

}

else

cnt=0;

}

}

void NRZL(int\* s,int size)

{

cout<<" Graphics part Not yet Done"<<endl;

return;

}

void NRZI(int\* s,int\* enco,int size,int size1)

{

cout<<"Graphics part Not yet Done"<<endl;

return;

}

void Manchester(int\* s,int size,int\* enco,int size1)

{

cout<<"Graphics part Not yet Done"<<endl;

return;

}

void DManchester(int\* s,int size,int\* enco,int size1)

{

cout<<"Graphics part Not yet Done"<<endl;

return;

}

// function generating a random sequence of binary data

int\* rand\_gen(int n)

{

int\* st=new int[n];

srand(time(0));

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

st[i]=rand()%2;

return st;

}

int main()

{

srand(time(0));

int ipt; //input

int entech; //Encoding Technique

string str;

int\* s;

int size;

cout<<endl;

cout<<"------------------ LINE ENCODING ---------------------\n";

cout<<endl;

cout<<"SELECT ONE OF THE OPTION FOR DATA GENERATION. "<<endl;

cout<<"\n";

cout<<"1 ~~ Random with some specific subsequences\n";

cout<<"\n";

cout<<"2 ~~ Complete Random \n";

cin>>ipt;

switch(ipt)

{

case 1:

{

cout<<"Enter your pattern (Binary Only)\n";

int s1 = 5+rand()%7;

int s2 = 5+rand()%7;

int \* x = rand\_gen(s1);

int \* y = rand\_gen(s2);

cin>>str;

size = s1+str.length()+s2;

s = new int[size];

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

{

if(i<s1)

s[i]=x[i];

else if(i<s1+str.length())

s[i]=str[i-s1]-'0';

else

s[i]=y[i-s1-str.length()];

}

cout<<endl;

break;

}

case 2:

int n=20+rand()%9;

s=rand\_gen(n);

size=n;

break;

}

cout<<"\nFOLLOWING WILL BE OUR INPUT:"<<endl;

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

cout<<s[i];

cout<<endl;

cout<<"\n";

cout<<"ENTER tYPE OF ENCODING FOR THE INPUT\n";

cout<<"\n";

cout<<"1- NRZ-L\n";

cout<<"\n";

cout<<"2- NRZ-I\n";

cout<<"\n";

cout<<"3- Manchester\n";

cout<<"\n";

cout<<"4- Differential Manchester\n";

cout<<"\n";

cin>>entech;

int \*enco;

switch(entech)

{

case 1:

{

NRZL(s,size);

break;

}

case 2:

{

enco = new int[size];

enco[0]=!s[0];

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

{

if(s[i]==1)

enco[i] =! enco[i-1];

else

enco[i] = enco[i-1];

}

for(int i=0; i<size; i++){

cout<<enco[i];

}

cout<<endl;

NRZI(s,enco,size,size);

break;

}

case 3:

{

enco = new int[2\*size];

for(int i=0,j=0;i<size;i++,j=j+2)

{

if(s[i])

{

enco[j]=1;

enco[j+1]=0;

}

else

{

enco[j]=0;

enco[j+1]=1;

}

}

for(int i=0; i<(2\*size); i++){

cout<<enco[i];

}

cout<<endl;

Manchester(s,size,enco,2\*size);

break;

}

case 4:

{

enco = new int[2\*size];

if(s[0])

{

enco[0]=1;

enco[1]=0;

}

else

{

enco[0]=0;

enco[1]=1;

}

for(int i=1,j=2;i<size;i++,j++)

{

if(s[i])

{

enco[j] = enco[j-1];

j++;

enco[j] =! enco[j-1];

}

else

{

enco[j] =! enco[j-1];

j++;

enco[j] =! enco[j-1];

}

}

for(int i=0; i<(2\*size); i++){

cout<<enco[i];

}

cout<<endl;

DManchester(s,size,enco,2\*size);

}

cout<<endl;

cout<<endl<<"THE END"<<endl;

}

cout<<"\n";

}