**CN PRACTICAL**

**2021337**

**Name: Lovely Rajput**

**1.Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.**

#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

int i,j,k,l;

int fs;

cout<<"\n Enter Frame size : ";

cin>>fs;

int f[20];

cout<<"\n Enter Frame : ";

for(i=0;i<fs;i++)

{

cin>>f[i];

}

int gs;

cout<<"\n Enter Generator size : ";

cin>>gs;

int g[20];

cout<<"\n Enter Generator : ";

for(i=0;i<gs;i++)

{

cin>>g[i];

}

cout<<"\n Sender Side : ";

cout<<"\n Frame: ";

for(i=0;i<fs;i++)

{

cout<<f[i];

}

cout<<"\n Generator : ";

for(i=0;i<gs;i++)

{

cout<<g[i];

}

int rs=gs-1;

cout<<"\n Number of 0's to be appended : "<<rs;

for (i=fs;i<fs+rs;i++)

{

f[i]=0;

}

int temp[20];

for(i=0;i<20;i++)

{

temp[i]=f[i];

}

cout<<"\n Message after appending 0's : ";

for(i=0; i<fs+rs;i++)

{

cout<<temp[i];

}

for(i=0;i<fs;i++)

{

j=0;

k=i;

{

for(j=0,k=i;j<gs;j++,k++)

{

if((temp[k]==1 && g[j]==1) || (temp[k]==0 &&

g[j]==0))

{

temp[k]=0;

}

else

{

temp[k]=1;

}

}

}

}

int crc[15];

for(i=0,j=fs;i<rs;i++,j++)

{

crc[i]=temp[j];

}

cout<<"\n CRC bits: ";

for(i=0;i<rs;i++)

{

cout<<crc[i];

}

cout<<"\n Transmitted Frame: ";

int tf[15];

for(i=0;i<fs;i++)

{

tf[i]=f[i];

}

for(i=fs,j=0;i<fs+rs;i++,j++)

{

tf[i]=crc[j];

}

for(i=0;i<fs+rs;i++)

{

cout<<tf[i];

}

cout<<"\n Receiver side : ";

cout<<"\n Received Frame: ";

for(i=0;i<fs+rs;i++)

{

cout<<tf[i];

}

for(i=0;i<fs+rs;i++)

{

temp[i]=tf[i];

}

for(i=0;i<fs+rs;i++)

{

j=0;

k=i;

if (temp[k]>=g[j])

{

for(j=0,k=i;j<gs;j++,k++)

{

if((temp[k]==1 && g[j]==1) || (temp[k]==0 &&

g[j]==0))

{

temp[k]=0;

}

else

{

temp[k]=1;

}

}

}

}

cout<<"\n Remainder: ";

int rrem[15];

for (i=fs,j=0;i<fs+rs;i++,j++)

{

rrem[j]= temp[i];

}

for(i=0;i<rs;i++)

{

cout<<rrem[i];

}

int flag=0;

for(i=0;i<rs;i++)

{

if(rrem[i]!=0)

{

flag=1;

}

}

if(flag==0)

{

cout<<"\n Since Remainder Is 0 Hence Message Transmitted From Sender To Receiver Is Correct";

}

else

{

cout<<"\n Since Remainder Is Not 0 Hence Message Transmitted From Sender To Receiver Contains Error";

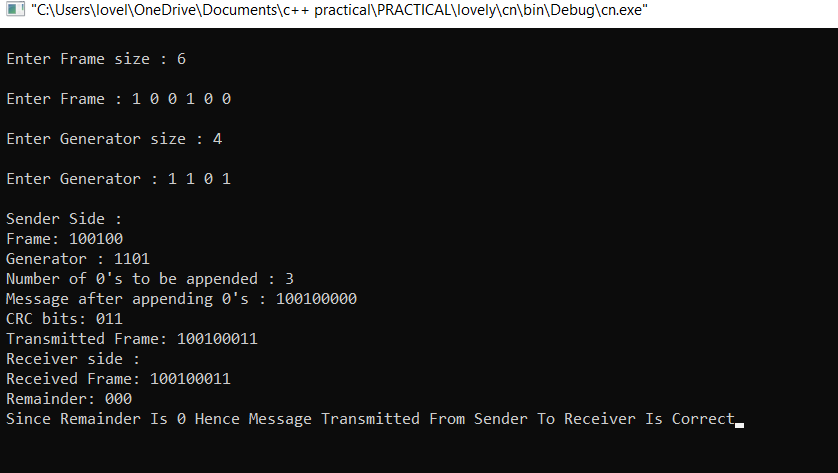
}

getch();

return 0;

}

**OUTPUT**



**2. Simulate and implement stop and wait protocol for noisy channel.**

#include<iostream>

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<direct.h>

using namespace std;

#define inc(k) if(k<max\_seq)k++;else k=0;

#define time 5

#define max\_seq 1

#define tot\_pack 5

int randn(int n)

{

return rand()%n + 1;

}

typedef struct

{

int data;

} packet;

typedef struct

{

int kind;

int seq;

int ack;

packet info;

}frame;

typedef enum

{

frame\_arrival,error,time\_out

}event\_type;

frame data1;

void from\_network\_layer(packet \*);

void to\_physical\_layer(frame \*);

void to\_network\_layer(packet \*);

void from\_physical\_layer(frame\*);

void sender();

void receiver();

void wait\_for\_event\_sender(event\_type \*);

void wait\_for\_event\_receiver(event\_type \*);

int i=1;

char turn;

int disc=0;

int main() {

while(!disc)

{ sender();

receiver();

}

getchar();

}

void sender()

{

static int frame\_to\_send=0;

static frame s;

packet buffer;

event\_type event;

static int flag=0;

if (flag==0)

{

from\_network\_layer(&buffer);

s.info=buffer;

s.seq=frame\_to\_send;

cout<<"\nSender information : " << s.info.data;

cout<<"\nSequence no. : " << s.seq;

turn='r';

to\_physical\_layer(&s);

flag=1;

}

wait\_for\_event\_sender(&event);

if(turn=='s')

{

if(event==frame\_arrival)

{

from\_network\_layer(&buffer);

inc(frame\_to\_send);

s.info=buffer;

s.seq=frame\_to\_send;

cout<<"\nSender information : " << s.info.data;

cout<<"\nSequence no. : " << s.seq;

getch();

turn='r';

to\_physical\_layer(&s);

}

}

}

void wait\_for\_event\_sender(event\_type \*e)

{

static int timer=0;

if(turn=='s')

{

timer++;

return ;

} else {

timer=0;

\*e=frame\_arrival;

}

}

void receiver()

{

static int frame\_expected=0;

frame s,r;

event\_type event;

wait\_for\_event\_receiver(&event);

if(turn=='r')

{

if(event==frame\_arrival)

{

from\_physical\_layer(&r);

if(r.seq==frame\_expected)

{

to\_network\_layer(&r.info);

inc (frame\_expected);

} else

cout<<"\nReceiver : Acknowledgement resent";

getch();

turn='s';

to\_physical\_layer(&s);

}

}

}

void to\_network\_layer(packet \*buffer)

{

cout<<"\nReceiver : packet received -> "<< i-1;

cout<<"\nAcknowledgement sent.";

getch();

if(i>tot\_pack)

{

disc=1;

cout<<"\nDiscontinue\n";

}

}

void from\_network\_layer(packet \*buffer) {

(\*buffer).data=i;

i++;

}

void wait\_for\_event\_receiver(event\_type \*e)

{

if(turn=='r') {

\*e=frame\_arrival;

}

}

void to\_physical\_layer(frame \*s) {

data1=\*s;

}

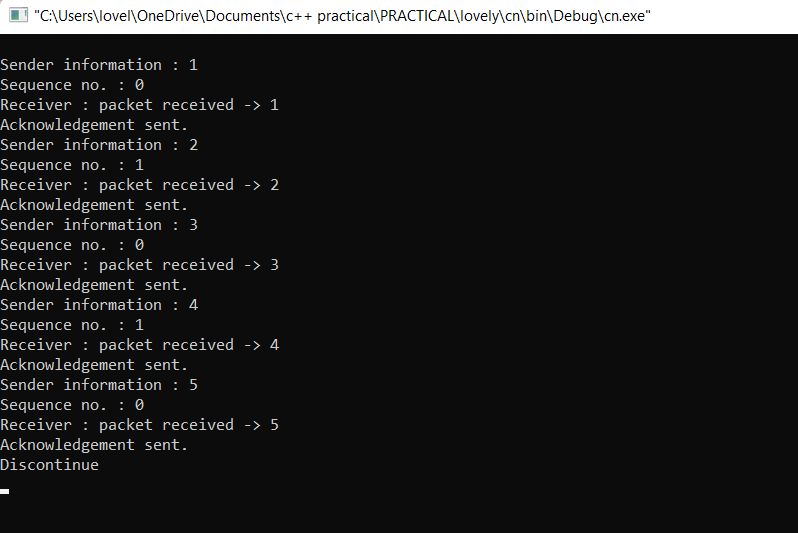
void from\_physical\_layer(frame \*buffer)

{

\*buffer=data1;

}

**OUTPUT**



**3. Simulate and implement go back n sliding window protocol.**

#include<stdio.h>

#include<conio.h>

int main()

{

char sender[50],receiver[50];

int i,winsize;

printf("\n Enter the window size : ");

scanf("%d",&winsize);

printf("\n Sender window is expanded to store message or window \n");

printf("\n Enter the data to be sent : ");

fflush(stdin);

gets(sender);

for(i=0;i<winsize;i++)

receiver[i]=sender[i];

receiver[i]=NULL;

printf("\n Message sent by the sender : ");

puts(sender);

printf("\n Window size of receiver is expanded\n");

printf("\n ACKNOWLEDGEMENT FROM RECEIVER \n");

for(i=0;i<winsize;i++);

printf("\n ACK:%d",i,"\n");

printf("\n Message received by receiver is : ");

puts(receiver);

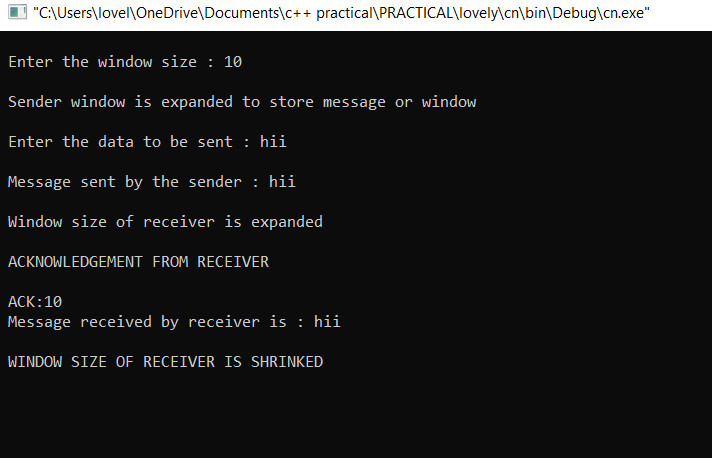
printf("\n WINDOW SIZE OF RECEIVER IS SHRINKED \n");

getch();

return 0;

}

**OUTPUT**



**4. Simulate and implement selective repeat sliding window protocol**.

#include<iostream>

#include<conio.h>

#include<stdlib.h>

#include<time.h>

#include<math.h>

using namespace std;

#define TOT\_FRAMES 500

#define FRAMES\_SEND 10

class sel\_repeat {

private:

int fr\_send\_at\_instance;

int arr[TOT\_FRAMES]; int

send[FRAMES\_SEND]; int

rcvd[FRAMES\_SEND]; char

rcvd\_ack[FRAMES\_SEND]; int

sw; int rw;

public:

void input(); void

sender(int); void

receiver(int);

};

void sel\_repeat::input() {

int n, m, i;

cout << "Enter the number of bits for the sequence no. :";

cin >> n; m = pow(2,n); int t = 0;

fr\_send\_at\_instance = (m/2);

for(i=0; i<TOT\_FRAMES;i++) {

arr[i] = t; t = (t+1)%m;

}

for(i=0; i<fr\_send\_at\_instance; i++)

{

send[i] = arr[i]; rcvd[i] = arr[i];

rcvd\_ack[i] = 'n';

}

rw = sw = fr\_send\_at\_instance;

sender(m);

}

void sel\_repeat::sender(int m) {

for(int i=0;i<fr\_send\_at\_instance; i++)

{ if(rcvd\_ack[i]=='n')

cout<<"SENDER : Frame "<<send[i]<<" is sent\n";

}

receiver(m);

}

void sel\_repeat::receiver(int m)

{ time\_t t; int f, j, f1, a1;

char ch;

srand((unsigned)time(&t));

for(int i=0;i<fr\_send\_at\_instance;i++) {

if(rcvd\_ack[i]=='n')

{

f=rand()%10; if(f!=5) {

for(int j=0;j<fr\_send\_at\_instance;j++)

if(rcvd[j]==send[i])

{

cout<<"receiver :Frame"<<rcvd[j]<<" received correctly\n"; rcvd[j] = arr[rw];

rw = (rw+1)%m;

break;

}

int j;

if(j==fr\_send\_at\_instance)

cout<<"receiver : Duplicate frame"<<send[i]<< "discarded\n";

a1 = rand()%5; if(a1==3)

{

cout << "(acknowledgment " <<send[i] << " lost)\n";

cout << "(sender timeouts ---> Resend the frame)\n";

rcvd\_ack[i] = 'n';

}

else {

cout<<"(acknowledgment " << send[i] << "received ) \n"; rcvd\_ack[i]='p';

}

}

else {

int ld=rand()%2;

if(ld==0) {

cout << "RECEIVER : Frame "<< send[i] << " is damaged\n"; cout << "RECEIVER : Negative Acknowledgment "<<send[i]<< " sent\n"; }

else { cout << "RECEIVER : Frame " << send[i] << "is lost\n"; cout << "(SENDER TIMEOUTS ---> RESEND THE FRAME)\n"; }

rcvd\_ack[i]='n'; }

} }

for(int j=0;j<fr\_send\_at\_instance;j++) {

if(rcvd\_ack[j]=='n')

break;

}

int i=0;

for(int k=j;k<fr\_send\_at\_instance;k++) {

send[i]=send[k];

if(rcvd\_ack[k]=='n')

rcvd\_ack[i]='n';

else rcvd\_ack[i]='p';

i++;

}

if(i!=fr\_send\_at\_instance) {

for(int k=i;k<fr\_send\_at\_instance;k++) {

send[k]=arr[sw];

sw=(sw+1)%m;

rcvd\_ack[k]='n';

}

}

cout << "Do you want to continue? (Press 'y'/'n') : ";

cin >> ch; cout << "\n";

if(ch=='y')

sender(m);

else exit(0);

}

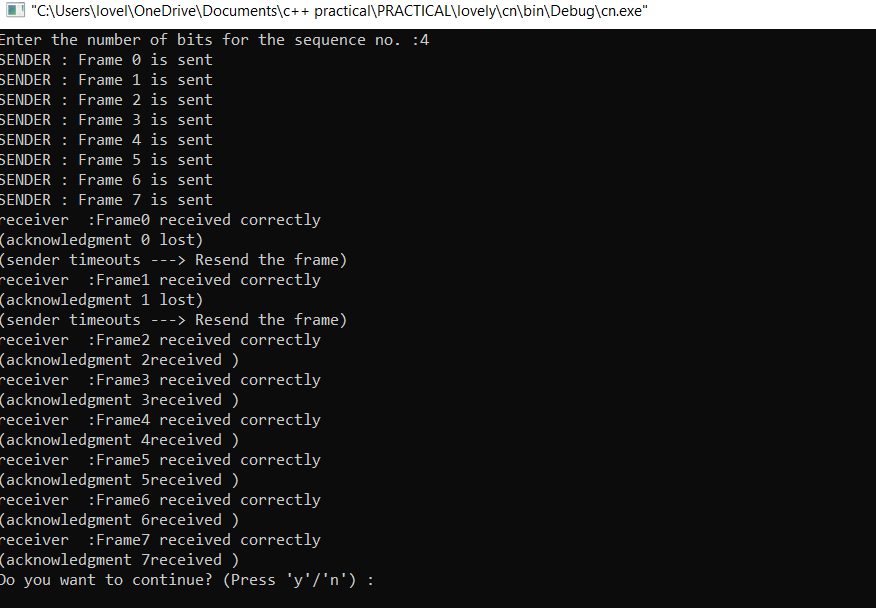
int main() {

sel\_repeat sr;

sr.input();

}

**OUTPUT**



**5. Simulate and implement distance vector routing algorithm**

#include<stdio.h>

#include<iostream>

using namespace std;

struct node

{

unsigned dist[6];

unsigned from[6];

}DVR[10];

int main()

{

cout<<"IMPLEMENTING DISTANCE VECTOR ROUTING ALGORITHM ";

int costmat[6][6];

int nodes, i, j, k;

cout<<"\nEnter the number of nodes : ";

cin>>nodes;

cout<<"\nEnter the cost matrix : \n" ;

for(i = 0; i < nodes; i++)

{

for(j = 0; j < nodes; j++)

{

cin>>costmat[i][j];

costmat[i][i] = 0;

DVR[i].dist[j] = costmat[i][j];

DVR[i].from[j] = j;

}

}

for(i = 0; i < nodes; i++)

for(j = i+1; j < nodes; j++)

for(k = 0; k < nodes; k++)

if(DVR[i].dist[j] > costmat[i][k] + DVR[k].dist[j])

{

DVR[i].dist[j] = DVR[i].dist[k] + DVR[k].dist[j];

DVR[j].dist[i] = DVR[i].dist[j];

DVR[i].from[j] = k;

DVR[j].from[i] = k;

}

for(i = 0; i < nodes; i++)

{

cout<<"\n\n For router: "<<i+1;

for(j = 0; j < nodes; j++)

cout<<"\t\n node "<<j+1<<" via "<<DVR[i].from[j]+1<<" Distance "<<DVR[i].dist[j];

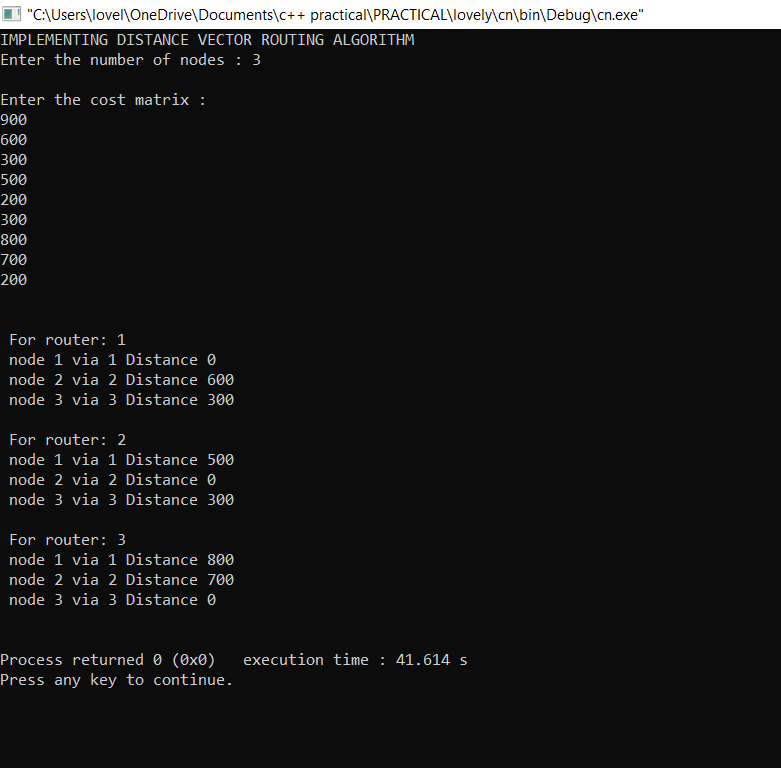
}

cout<<" \n\n ";

return 0;

}

**OUTPUT**



**6. Simulate and implement Dijkstra algorithm for shortest path routing.**

#include<iostream>

#include<stdio.h>

using namespace std;

#define INFINITY 9999

#define max 5

void dijkstra(int G[max][max],int n,int startnode);

int main() {

int

G[max][max]={{0,1,0,3,10},{1,0,5,0,0},{0,5,0,2,1},{3,0,2,0,6},{10,0,1,6,0}};

int n=5;

int u=0;

dijkstra(G,n,u);

return 0;

}

void dijkstra(int G[max][max],int n,int startnode) {

int cost[max][max],distance[max],pred[max];

int visited[max],count,mindistance,nextnode,i,j;

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

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

if(G[i][j]==0)

cost[i][j]=INFINITY;

else

cost[i][j]=G[i][j];

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

distance[i]=cost[startnode][i];

pred[i]=startnode;

visited[i]=0;

}

distance[startnode]=0;

visited[startnode]=1;

count=1;

while(count<n-1) {

mindistance=INFINITY;

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

if(distance[i]<mindistance&&!visited[i]) {

mindistance=distance[i];

nextnode=i;

}

visited[nextnode]=1;

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

if(!visited[i])

if(mindistance+cost[nextnode][i]<distance[i]) {

distance[i]=mindistance+cost[nextnode][i];

pred[i]=nextnode;

}

count++;

}

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

if(i!=startnode) {

cout<<"\nDistance of node"<<i<<"="<<distance[i];

cout<<"\nPath="<<i;

j=i;

do {

j=pred[j];

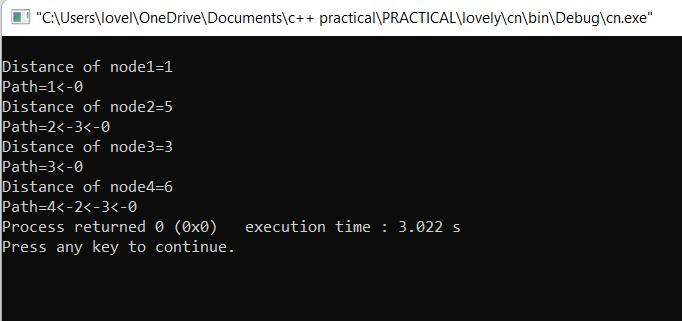
cout<<"<-"<<j;

}while(j!=startnode);

}

}

**OUTPUT**



**7.Write a program in C++ to implement Hamming code.**

#include <iostream>

#include <math.h>

#include <cstring>

using namespace std;

int checkParity(int n,int i,int code[])

{

int p=0,k;

for(int j=i;j<=n;j=k+i)

{

for(k=j;k<j+i && k<=n;k++)

{

if(code[k]==1)

p++;

}

}

if(p%2==0)

return 0;

else

return 1;

}

void hamming\_code(int data[], int num)

{

int r=0,m=0,n,j=1,c,code[50];

while((r+num+1)>(pow(2,r)))

r++;

n=num+r;

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

{

if(i==pow(2,m) && m<=r)

{

code[i]=0;

m++;

}

else

{

code[i]=data[j];

j++;

}

}

m=0;

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

{

if(i==pow(2,m) && m<=r)

{

c=checkParity(n,i,code);

code[i]=c;

m++;

}

}

cout<<"The hamming code for given data is: ";

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

cout<<code[i];

cout<<"\nEnter the received code: ";

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

cin>>code[i];

m=j=c=0;

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

{

if(i==pow(2,m) && m<=r)

{

c=c+(pow(2,j)\*checkParity(n,i,code));

j++;

m++;

}

}

if(c==0)

cout<<"\nReceived word is correct.";

else

{

cout<<"\nThere is error in bit "<<(n-c)+1<<"\nThe corrected code is:";

if(code[c]==1)

code[c]=0;

else

code[c]=1;

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

cout<<code[i];

}

}

int main()

{

int data[50], num;

cout<<"Enter the size of data: ";

cin>>num;

cout<<"Enter the data:";

for(int i=num;i>0;i--)

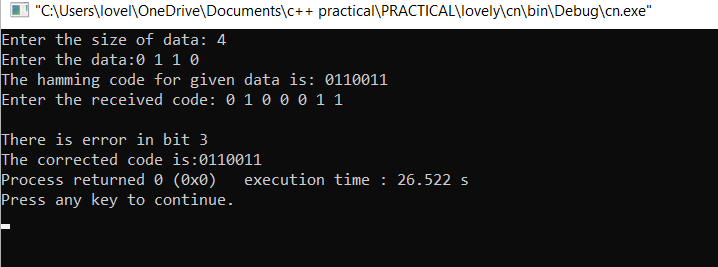
cin>>data[i];

hamming\_code(data, num);

return 0;

}

**OUTPUT**

****