NAME : KHUSHI PANWAR
ROLL NO: 2021334
COMPUTER NETWORKS PRACTICAL FILE

1. **Write a program to implement Cyclic Redundancy check (CRC) algorithm for noisy channel :**

```
#include <iostream>
#include <stdlib.h>
using namespace std;

class CRC_error_detection
{
public:
    int n, d[15], g[10], k, m;
    int rem[20];
    string str;

public:
    void input();
    void coffecient();
    void codeword();
    int exor(int, int);
    void modulus_2_division();
    void append();
};
void CRC_error_detection ::input()
{
    cout << "-> Enter the number of bits in dataword :";
    cin >> n;
    cout << "Enter the dataword :";
    for (int i = 0; i < n; i++)
    {
        cin >> d[i];
    }
    cout << "Enter the polynomial :";
    cin >> str;
    coffecient();
}
void CRC_error_detection ::coffecient()
{
    k = 0;
    for (int i = 0; i < str.size(); i++)
    {
        if (i == 0)
        {
            if (str.substr(i, 1) == "x")
```

```
            {
                g[k] = 1;
                k++;
            }
        }
        if (str.substr(i, 1) == "+")
        {
            if (str.substr(i + 1, 1) != "x")
            {
                g[k] = 0;
                k++;
            }
            else
            {
                g[k] = 1;
                k++;
            }
        }
    }
    cout << "CRC generator is :";
    for (int i = 0; i < k; i++)
    {
        cout << g[i] << " ";
    }
    codeword();
}
void CRC_error_detection ::codeword()
{
    m = n + (k - 1);
    cout << endl
         << "Code word is :";
    for (int i = n; i < m; i++)
    {
        d[i] = 0;
    }
    for (int i = 0; i < m; i++)
    {
        cout << d[i] << " ";
    }
}
int CRC_error_detection ::exor(int a, int b)
{
    if (a == b)
        return 0;
    else
        return 1;
}
void CRC_error_detection ::modulus_2_division()
{
```

```
    int temp[20];
    for (int i = 0; i < k; i++)
        rem[i] = d[i];
    for (int j = k; j <= m; j++)
    {
        for (int i = 0; i < k; i++)
            temp[i] = rem[i];
        if (rem[0] == 0)
        {
            for (int i = 0; i < k - 1; i++)
                rem[i] = temp[i + 1];
        }
        else
        {
            for (int i = 0; i < k - 1; i++)
                rem[i] = exor(temp[i + 1], g[i + 1]);
        }
        if (j != m)
            rem[k - 1] = d[j];
        else
            rem[k - 1] = 0;
    }
}
void CRC_error_detection ::append()
{
    cout << endl
         << "CRC :";
    for (int i = 0; i < k - 1; i++)
    {
        cout << rem[i] << " ";
    }
    int p = 0;
    for (int i = n; i < m; i++)
    {
        d[i] = rem[p];
        p++;
    }
    cout << endl
         << "Transmitted data is :";
    for (int i = 0; i < m; i++)
    {
        cout << d[i] << " ";
    }
    cout << endl
         << "\n*Data recieved by the receiver*" << endl;
    string ch;
    cout << "\nEnter the type :";
    cin >> ch;
    if (ch == "noisy" || ch == "Noisy")
```

```
    {
        int s;
        srand(0);
        s = rand() % m;
        cout << "Before Codeword[" << s << "]=" << d[s] << endl;
        if (d[s] == 0)
            d[s] = 1;
        else
            d[s] = 0;
        cout << "Now Codeword[" << s << "]=" << d[s];
    }
    cout << "\nNow Codeword becomes :";
    for (int i = 0; i < m; i++)
    {
        cout << d[i] << " ";
    }
    modulus_2_division();
    cout << "\nRemainder :";
    for (int i = 0; i < k - 1; i++)
    {
        cout << rem[i] << " ";
    }
    cout << endl;
    int flag = 0;
    for (int i = 0; i < k - 1; i++)
    {
        if (rem[i] != 0)
        {
            flag = 1;
            break;
        }
    }
    if (flag == 0)
        cout << "\n received correct data .";
    else
        cout << "\nsome error in data please check and send again.";
}
int main()
{
    cout<<endl<<"\t ** PROGRAM TO IMPLEMENT CYCLIC RENUNDANCY CHECK
** "<<endl<<endl;

    CRC_error_detection obj;
    obj.input();
    obj.modulus_2_division();
    obj.append();
    return 0;
}
```

```
          ** PROGRAM TO IMPLEMENT CYCLIC RENUNDANCY CHECK **

  -> Enter the number of bits in dataword :3
  Enter the dataword :1 0 1
  Enter the polynomial :2x^2+3x+4
  CRC generator is :0 0
  Code word is :1 0 1 0
  CRC :0
  Transmitted data is :1 0 1 0

  *Data recieved by the receiver*

  Enter the type :noisy
  Before Codeword[2]=1
  Now Codeword[2]=0
  Now Codeword becomes :1 0 0 0
  Remainder :0

   received correct data .
  PS E:\_Bsc CS\3rd Sem Practical>
```

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

```cpp
#include <iostream>
#include <time.h>
#include <cstdlib>
#include <ctime>
#include <unistd.h>
using namespace std;
class timer
{
private:
    unsigned long begTime;

public:
    void start()
    {
        begTime = clock();
    }
    unsigned long elapsedTime()
    {
        return ((unsigned long)clock() - begTime) / CLOCKS_PER_SEC;
```

```
    }
    bool isTimeout(unsigned long seconds)
    {
        return seconds >= elapsedTime();
    }
};
int main()
{
    cout << endl<< "\t ** PROGRAM TO IMPLEMENT STOP AND WAIT PROTOCOL ** "
<< endl << endl;
    int fcount;
    cout<<"-> Enter the number of frames : ";
    cin>>fcount;
    cout<<"-> Enter the frames : ";
    int frames[10];
    for (int i=0 ; i<fcount; i++) {
        cin>>frames[i];
    }
    unsigned long seconds = 5;
    srand(time(NULL));
    timer t;
    cout << "Sender has to send frames : ";
    for (int i = 0; i < fcount; i++)
        cout << frames[i] << " ";
    cout << endl;
    int count = 0;
    bool delay = false;
    cout << endl
         << "Sender\t\t\t\t\tReceiver" << endl;
    do
    {
        bool timeout = false;
        cout << "Sending Frame : " << frames[count];
        cout.flush();
        cout << "\t\t";
        t.start();
        if (rand() % 2)
        {
            int to = 24600 + rand() % (64000 - 24600) + 1;
            for (int i = 0; i < 64000; i++)
                for (int j = 0; j < to; j++)
                {
                }
        }
        if (t.elapsedTime() <= seconds)
        {
            cout << "Received Frame : " << frames[count] << " ";
            if (delay)
            {
                cout << "Duplicate";
                delay = false;
```

```
            }
            cout << endl;
            count++;
        }
        else
        {
            cout << "---" << endl;
            cout << "Timeout" << endl;
            timeout = true;
        }
        t.start();
        if (rand() % 2 || !timeout)
        {
            int to = 24600 + rand() % (64000 - 24600) + 1;
            for (int i = 0; i < 64000; i++)
                for (int j = 0; j < to; j++)
                {
                }
            if (t.elapsedTime() > seconds)
            {
                cout << "Delayed Ack" << endl;
                count--;
                delay = true;
            }
            else if (!timeout)
                cout << "Acknowledgement : " << frames[count] - 1 << endl;
        }
    } while (count != fcount);
    return 0;
}
```

```
          ** PROGRAM TO IMPLEMENT STOP AND WAIT PROTOCOL **

   -> Enter the number of frames : 5
   -> Enter the frames : 1 2 3 4 5
   Sender has to send frames : 1 2 3 4 5

   Sender                                  Receiver
   Sending Frame : 1                Received Frame : 1
   Acknowledgement : 1
   Sending Frame : 2                Received Frame : 2
   Acknowledgement : 2
   Sending Frame : 3                Received Frame : 3
   Acknowledgement : 3
   Sending Frame : 4                Received Frame : 4
   Delayed Ack
   Sending Frame : 4                Received Frame : 4 Duplicate
   Acknowledgement : 4
   Sending Frame : 5                Received Frame : 5
   Delayed Ack
   Sending Frame : 5                Received Frame : 5 Duplicate
   Delayed Ack
   Sending Frame : 5                Received Frame : 5 Duplicate
   Acknowledgement : 32758
   PS E:\_Bsc CS\3rd Sem Practical>
```

3. **Simulate and implement the Go back and sliding window protocol :**

```cpp
#include<iostream>
using namespace std;

int main()
{
     cout << endl<< "\t ** PROGRAM TO IMPLEMENT SLIDING WINDOW
PROTOCOL ** " << endl << endl;

    int w,i,f,frames[50];

    cout<<"Enter window size: ";
    cin>>w;

    cout<<"\nEnter number of frames to transmit: ";
    cin>>f;

    cout<<"\nEnter "<<f<<" frames: ";

    for(i=1;i<=f;i++)
        cin>>frames[i];

    cout<<"\nWith sliding window protocol the frames will be sent in
the following manner (assuming no corruption of frames)\n\n";
```

```
    cout<<"After sending "<<w<<" frames at each stage sender waits
for acknowledgement sent by the receiver\n\n";

    for(i=1;i<=f;i++)
    {
        if(i%w==0)
        {
            cout<<frames[i]<<"\n";
            cout<<"Acknowledgement of above frames sent is received
by sender\n\n";
        }
        else
            cout<<frames[i]<<" ";
    }

    if(f%w!=0)
        cout<<"\nAcknowledgement of above frames sent is received by
sender\n";

    return 0;
}
```

```
          ** PROGRAM TO IMPLEMENT SLIDING WINDOW PROTOCOL **

  Enter window size: 3

  Enter number of frames to transmit: 6

  Enter 6 frames: 1 2 3 4 5 6

  With sliding window protocol the frames will be sent in the following manner (assuming no cor
  ruption of frames)

  After sending 3 frames at each stage sender waits for acknowledgement sent by the receiver

  1 2 3
  Acknowledgement of above frames sent is received by sender

  4 5 6
  Acknowledgement of above frames sent is received by sender

  PS E:\_Bsc CS\3rd Sem Practical>
```

4. **Simulate and implement the 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;
    int m;
    int i;
    cout << "*****Please enter the no. 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)
{
    cout << "----------------------------------------------------
-" << endl;
```

```
    for (int i = 0; i < fr_send_at_instance; i++)
    {
        if (rcvd_ack[i] == 'n')
            cout << "SENDER : Frame " << send[i] << "
is============> SENT!!\n";
    }
    cout << "\n-----------------------------------------------------"
<< endl;
    receiver(m);
}
void sel_repeat::receiver(int m)
{
    time_t t;
    int f;
    int j;
    int f1;
    int 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 << "reciever:Frame" << rcvd[j] <<
"recieved correctly\n";
                        rcvd[j] = arr[rw];
                        rw = (rw + 1) % m;
                        break;
                    }
                int j;
                if (j == fr_send_at_instance)
                    cout << "reciever:Duplicate frame" << send[i] <<
"discarded\n";
                a1 = rand() % 5;
                if (a1 == 3)
                {
                    cout << "(acknowledgement " << send[i] << "
lost)\n";
                    cout << "(sender timeouts-->Resend the
frame)\n";
                    rcvd_ack[i] = 'n';
                }
                else
```

```
                {
                    cout << "(acknowledgement " << send[i] << "
recieved)\n";
                    rcvd_ack[i] = 'p';
                }
            }
            else
            {
                int ld = rand() % 2;
                if (ld == 0)
                {
                    cout << "RECEIVER : Frame " << send[i] << " is
damaged\n";
                    cout << "RECEIVER : Negative Acknowledgement "
<< 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?[Y/n]";
    cin >> ch;
    cout << "\n";
    if (ch == 'y')
        sender(m);
    else
        exit(0);
}
int main()
{
    cout << endl<< "\t ** PROGRAM TO IMPLEMENT SELECTIVE REPEAT
SLIDING WINDOW PROTOCOL ** " << endl << endl;
    sel_repeat sr;
    sr.input();
    return 0;
}
```

```
          ** PROGRAM TO IMPLEMENT SELECTIVE REPEAT SLIDING WINDOW PROTOCOL **

   *****Please enter the no. of bits for the sequence no.**** : 3
   ------------------------------------------------------
   SENDER : Frame 0 is============> SENT!!
   SENDER : Frame 1 is============> SENT!!
   SENDER : Frame 2 is============> SENT!!
   SENDER : Frame 3 is============> SENT!!

   ------------------------------------------------------
   reciever:Frame0recieved correctly
   (acknowledgement 0 recieved)
   reciever:Frame1recieved correctly
   (acknowledgement 1 recieved)
   reciever:Frame2recieved correctly
   (acknowledgement 2 recieved)
   reciever:Frame3recieved correctly
   (acknowledgement 3 lost)
   (sender timeouts-->Resend the frame)
   Do you want to continue?[Y/n]n

   PS E:\_Bsc CS\3rd Sem Practical>
```

5. **Simulate and implement the selective repeat sliding window protocol :**

```
#include<stdio.h>
#include<iostream>
```

```
using namespace std;
struct node
{
    unsigned dist[6];
    unsigned from[6];
}DVR[10];
int main()
{
    cout << endl<< "\t ** PROGRAM TO IMPLEMENT DISTANCE VECTOR
ROUTING ALGORITHM ** " << endl << endl;
    int costmat[6][6];
    int nodes, i, j, k;
    cout<<"-> Enter the number of nodes : ";
    cin>>nodes; //Enter the nodes
    cout<<"-> Enter 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]; //initialise the
distance equal to cost matrix
            DVR[i].from[j] = j;
        }
    }
            for(i = 0; i < nodes; i++) //We choose arbitary vertex k
and we calculate the direct distance from the node i to k using the
cost matrix and add the distance from k to node j
            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])
                {   //We calculate the minimum distance
                    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;
}
```

```
        ** PROGRAM TO IMPLEMENT DISTANCE VECTOR ROUTING ALGORITHM **

  -> Enter the number of nodes : 3
  -> Enter the cost matrix :
  12 4 7
  6 4 9
  4 6 9


   For router: 1
   node 1 via 1 Distance 0
   node 2 via 2 Distance 4
   node 3 via 3 Distance 7

   For router: 2
   node 1 via 1 Distance 6
   node 2 via 2 Distance 0
   node 3 via 3 Distance 9

   For router: 3
   node 1 via 1 Distance 4
   node 2 via 2 Distance 6
   node 3 via 3 Distance 0
```

6. **Simulate and implement the Dijkstra Algorithm for shortest path routing :**

```cpp
#include<iostream>
#include<iomanip>
using namespace std;

int miniDist(int distance[], bool Tset[]) // finding minimum
distance
{
    int minimum=INT_MAX,ind;

    for(int k=0;k<6;k++)
    {
        if(Tset[k]==false && distance[k]<=minimum)
        {
            minimum=distance[k];
            ind=k;
        }
    }
    return ind;
```

```
}

void DijkstraAlgo(int graph[10][10],int src, int size) // adjacency
matrix
{
    int distance[10]; // // array to calculate the minimum distance
for each node
    bool Tset[10];// boolean array to mark visited and unvisited for
each node


    for(int k = 0; k<size; k++)
    {
        distance[k] = INT_MAX;
        Tset[k] = false;
    }

    distance[src] = 0;   // Source vertex distance is set
0

    for(int k = 0; k<6; k++)
    {
        int m=miniDist(distance,Tset);
        Tset[m]=true;
        for(int k = 0; k<6; k++)
        {
            // updating the distance of neighbouring vertex
            if(!Tset[k] && graph[m][k] && distance[m]!=INT_MAX &&
distance[m]+graph[m][k]<distance[k])
                distance[k]=distance[m]+graph[m][k];
        }
    }
    cout<<"Vertex\t\tDistance from source vertex"<<endl;
    for(int k = 0; k<6; k++)
    {
        char str=65+k;
        cout<<str<<"\t\t\t"<<distance[k]<<endl;
    }
}

void input(int A[10][10], int size){
    cout<<"-> Enter the graph here : "<<endl;
    for (int i=0; i<size; i++){
        for (int j=0; j<size; j++){
            cin>>A[i][j];
        }
    }
}
```

```
int main()
{
    cout << endl<< "\t ** PROGRAM TO IMPLEMENT SELECTIVE REPEAT
SLIDING WINDOW PROTOCOL ** " << endl << endl;
    int size;
    cout<<"-> Enter the size of the graph : ";
    cin>>size;
    int graph[10][10];
    input(graph, size);
    DijkstraAlgo(graph,0, size);
    return 0;
}
```

```
        ** PROGRAM TO IMPLEMENT SELECTIVE REPEAT SLIDING WINDOW PROTOCOL **

 -> Enter the size of the graph : 6
 -> Enter the graph here :
 0 1 2 0 0 0
 1 0 0 5 1 0
 2 0 0 2 3 0
 0 5 2 0 2 2
 0 1 3 2 0 1
 0 0 0 2 1 0
 Vertex          Distance from source vertex
 A                       0
 B                       1
 C                       2
 D                       4
 E                       2
 F                       3
 PS E:\_Bsc CS\3rd Sem Practical>
```

7. **Write a program to implement Hamming code :**

```
#include <iostream>
#include <cmath>
#include <string>
using namespace std;
class Hamming
{
    string message;
    int codeword[50], temp[50];
    int n, check;
    char parity;
```

```
public:
    Hamming()
    {
        parity = 'E';
        message = "";
        n = check = 0;
        for (int i = 0; i < 50; i++)
        {
            temp[i] = codeword[i] = 0;
        }
    }
    void generate()
    {
        do
        {
            cout << "Enter the message in binary : ";
            cin >> message;
        } while (message.find_first_not_of("01") != string::npos);
        n = message.size();
        cout << "Odd(O)/Even(E) Parity ? ";
        cin >> parity;
        for (unsigned int i = 0; i < message.size(); i++)
        {
            if (message[i] == '1')
                temp[i + 1] = 1;
            else
                temp[i + 1] = 0;
        }
        computeCode();
    }
    void computeCode()
    {
        check = findr();
        cout << "Number of Check Bits : " << check << endl;
        cout << "Number of Bits in Codeword : " << n + check <<
endl;
        for (int i = (n + check), j = n; i > 0; i--)
        {
            if ((i & (i - 1)) != 0)
                codeword[i] = temp[j--];
            else
                codeword[i] = setParity(i);
        }
        cout << "Parity Bits - ";
        for (int i = 0; i < check; i++)
            cout << "P" << pow(2, i) << " : " <<
codeword[(int)pow(2, i)] << "\t";
        cout << endl;
        cout << "Codeword :" << endl;
```

```
        for (int i = 1; i <= (n + check); i++)
            cout << codeword[i] << " ";
        cout << endl;
    }
    int findr()
    {
        for (int i = 1;; i++)
        {
            if (n + i + 1 <= pow(2, i))
                return i;
        }
    }
    int setParity(int x)
    {
        bool flag = true;
        int bit;
        if (x == 1)
        {
            bit = codeword[x + 2];
            for (int j = x + 3; j <= (n + check); j++)
            {
                if (j % 2)
                {
                    bit ^= codeword[j];
                }
            }
        }
        else
        {
            bit = codeword[x + 1];
            for (int i = x; i <= (n + check); i++)
            {
                if (flag)
                {
                    if (i == x || i == x + 1)
                        bit = codeword[x + 1];
                    else
                        bit ^= codeword[i];
                }
                if ((i + 1) % x == 0)
                    flag = !flag;
            }
        }
        if (parity == 'O' || parity == 'o')
            return !bit;
        else
            return bit;
    }
    void correct()
```

```
{
    do
    {
        cout << "Enter the received codeword : ";
        cin >> message;
    } while (message.find_first_not_of("01") != string::npos);
    for (unsigned int i = 0; i < message.size(); i++)
    {
        if (message[i] == '1')
            codeword[i + 1] = 1;
        else
            codeword[i + 1] = 0;
    }
    detect();
}
void detect()
{
    int position = 0;
    cout << "Parity Bits - ";
    for (int i = 0; i < check; i++)
    {
        bool flag = true;
        int x = pow(2, i);
        int bit = codeword[x];
        if (x == 1)
        {
            for (int j = x + 1; j <= (n + check); j++)
            {
                if (j % 2)
                {
                    bit ^= codeword[j];
                }
            }
        }
        else
        {
            for (int k = x + 1; k <= (n + check); k++)
            {
                if (flag)
                {
                    bit ^= codeword[k];
                }
                if ((k + 1) % x == 0)
                    flag = !flag;
            }
        }
        cout << "P" << x << ": " << bit << "\t";
        if ((parity == 'E' || parity == 'e') && bit == 1)
            position += x;
```

```
            if ((parity == 'O' || parity == 'o') && bit == 0)
                position += x;
        }
        cout << endl
             << "Received Codeword :" << endl;
        for (int i = 1; i <= (n + check); i++)
            cout << codeword[i] << " ";
        cout << endl;
        if (position != 0)
        {
            cout << "Error at bit : " << position << endl;
            codeword[position] = !codeword[position];
            cout << "Corrected Codeword : " << endl;
            for (int i = 1; i <= (n + check); i++)
                cout << codeword[i] << " ";
            cout << endl;
        }
        else
            cout << "No Error in Received code." << endl;
        cout << "Received Message is : ";
        for (int i = 1; i <= (n + check); i++)
            if ((i & (i - 1)) != 0)
                cout << codeword[i] << " ";
        cout << endl;
    }
};
int main()
{
    cout << endl<< "\t ** PROGRAM TO IMPLEMENT HAMMING CODE ** " <<
endl << endl;
    char choice;
    do
    {
        Hamming a;
        cout << "At Sender's side : " << endl;
        a.generate();
        cout << endl
             << "At Receiver's Side : " << endl;
        a.correct();
        cout << endl
             << "Enter another code ? (Y/N) : ";
        cin >> choice;
        cout << endl;
    } while (choice == 'y' || choice == 'Y');
    return 0;
}
```

```
        ** PROGRAM TO IMPLEMENT HAMMING CODE **

At Sender's side :
Enter the message in binary : 101100
Odd(O)/Even(E) Parity ? E
Number of Check Bits : 4
Number of Bits in Codeword : 10
Parity Bits - P1 : 0    P2 : 1  P4 : 0  P8 : 0
Codeword :
0 1 1 0 0 1 1 0 0 0

At Receiver's Side :
Enter the received codeword : 101101
Parity Bits - P1: 1     P2: 1   P4: 1   P8: 0
Received Codeword :
1 0 1 1 0 1 1 0 0 0
Error at bit : 7
Corrected Codeword :
1 0 1 1 0 1 0 0 0 0
Received Message is : 1 0 1 0 0 0

Enter another code ? (Y/N) : n

PS E:\_Bsc CS\3rd Sem Practical>
```

```
        ** PROGRAM TO IMPLEMENT HAMMING CODE **

At Sender's side :
Enter the message in binary : 101101
Odd(O)/Even(E) Parity ? O
Number of Check Bits : 4
Number of Bits in Codeword : 10
Parity Bits - P1 : 1    P2 : 1  P4 : 1  P8 : 0
Codeword :
1 1 1 1 0 1 1 0 0 1

At Receiver's Side :
Enter the received codeword : 101100
Parity Bits - P1: 1     P2: 1   P4: 0   P8: 1
Received Codeword :
1 0 1 1 0 0 1 0 0 1
Error at bit : 4
Corrected Codeword :
1 0 1 0 0 0 1 0 0 1
Received Message is : 1 0 0 1 0 1

Enter another code ? (Y/N) :
```