Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403

Assignment-03

📝 Experiment -01

🎯 AIM: Write a program for hamming code error detection and correction.

✔ Solution :

```
#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::npo
s);
        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::npo
s);
        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()
{
    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;
}
```

Output ScreenShot 🖼 :

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\CP-Code> E:\CP-Code\practice.exe
At Sender's side :
Enter the message in binary : 1011
Odd(O)/Even(E) Parity ? o
Number of Check Bits : 3
Number of Bits in Codeword : 7
Parity Bits - P1 : 1    P2 : 0  P4 : 1
Codeword :
1 0 1 1 0 1 1

At Receiver's Side :
Enter the received codeword : 10110100
Parity Bits - P1: 0     P2: 0   P4: 0
Received Codeword :
1 0 1 1 0 1 0
Error at bit : 7
Corrected Codeword :
1 0 1 1 0 1 1
Received Message is : 1 0 1 1

Enter another code ? (Y/N) : N

PS E:\CP-Code>
```

Experiment -02

🎯 AIM: Write a program to implement Stop & wait ARQ protocol.

✔ Solution :

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
using namespace std;
#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;
//creating prototype
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 *);
//end
#define inc(k)       \
    if (k < max_seq) \
        k++;         \
    else             \
        k = 0;
int i = 1;
char turn;
int disc = 0;
int main()
{
    while (!disc)
    {
        sender();
        // delay(400);
        receiver();
    }
    getchar();
}
void sender()
{
    static int frame_to_send = 0;
    static frame s;
    packet buffer;
```

```
    event_type event;
    static int flag = 0; //first place
    if (flag == 0)
    {
        from_network_layer(&buffer);
        s.info = buffer;
        s.seq = frame_to_send;
        cout << "\nsender information \t" << s.info.data << "\n
";
        cout << "\nsequence no. \t" << 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 \t" << s.info.data <<
 "\n";
            cout << "\nsequence no. \t" << s.seq << "\n";
            getch();
            turn = 'r';
            to_physical_layer(&s);
        }
    }
} //end of sender function
void from_network_layer(packet *buffer)
{
    (*buffer).data = i;
    i++;
```

```
} //end of from network layer function
void to_physical_layer(frame *s)
{
    data1 = *s;
} //end of to physical layer function
void wait_for_event_sender(event_type *e)
{
    static int timer = 0;
    if (turn == 's')
    {
        timer++;
        //timer=0;
        return;
    }
    else //event is frame arrival
    {
        timer = 0;
        *e = frame_arrival;
    }
} //end of wait for event function
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 \n"
;
            getch();
            turn = 's';
            to_physical_layer(&s);
        }
    }
} //end of receiver function
void wait_for_event_receiver(event_type *e)
{
    if (turn == 'r')
    {
        *e = frame_arrival;
    }
}
void from_physical_layer(frame *buffer)
{
    *buffer = data1;
}
void to_network_layer(packet *buffer)
{
    cout << "\nReceiver : packet received \t" << i - 1;
    cout << "\n Acknowledgement sent \t";
    getch();
    if (i > tot_pack)
    {
        disc = 1;
        cout << "\ndiscontinue\n";
    }
}
```

```
PS E:\CP-Code> E:\CP-Code\practice.exe

sender information      1

sequence no.    0
Receiver : packet received      1
 Acknowledgement sent
sender information      2

sequence no.    1

Receiver : packet received      2
 Acknowledgement sent
sender information      3

sequence no.    0

Receiver : packet received      3
 Acknowledgement sent
sender information      4

sequence no.    1

Receiver : packet received      4
 Acknowledgement sent
sender information      5

sequence no.    0

Receiver : packet received      5
 Acknowledgement sent
discontinue
```

---- END OF ASSIGNMENT ----