UNIVERSITY OF DELHI

COLLEGE OF VOCATIONAL STUDIES

BSC (HONS) COMPUTER SCIENCE

SEMESTER - 3

COMPUTER NETWORKS

ANUJ KUMAR SEN

2K21/CS/15

Question 1 :

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

Solution 1 :

```
/* Created By - ANUJ KUMAR SEN */

#include <iostream>
#include <string>
using namespace std;

/* To remove leading spaces in a string */
string removeLeading0s(string str)
{
    int i;
    for (i = 0; i < str.length(); i++)
        if (str[i] == '1')
        {
            break;
        }
    return str.substr(i);
}
/* To check if each bit in string is zero */
bool isZero(string str)
{
    str = removeLeading0s(str);
    if (str.length() <= 0)
        return true;
    else
        return false;
}
/* To check if input string has any errors or illegal characters */
bool checkInput(string str)
{
    str = removeLeading0s(str);
    if (isZero(str))
    {
        cout << "\nInput Error!\nEntered bits are only zero.\n"
             << endl;
        return false;
    }
    for (int i = 0; i < str.length(); ++i)
    {
        if (str[i] != '1' && str[i] != '0')
```

```
        {
            cout << "\nInput Error!\nEntered bits contain characters other than 0
or 1.\n"
                 << endl;
            return false;
        }
    }
    return true;
}

string division(string dividend, string divisor)
{
    string remainder = "";
    for (int i = 0; i <= dividend.length() - divisor.length();)
    {
        /* Taking XOR, bit by bit */
        for (int j = 0; j < divisor.length(); ++j)
            dividend[i + j] = (dividend[i + j] == divisor[j]) ? '0' : '1';

        /* To find the index for next iter*/
        while (i < dividend.length() && dividend[i] != '1')
            ++i;
    }
    /*
        Calculating remainder i.e. last n-1 digits of encoded string, n: length of
gen
    */
    remainder = dividend.substr(dividend.length() - divisor.length());
    return remainder;
}

void CRC(string encoded, string gen)
{
    int i;
    /* Checking for error by again performing modulo 2 division */
    cout << "\nApplying Cyclic Redundancy Check...." << endl;
    string rem = division(encoded, gen);
    if (!isZero(rem))
        cout << "\nError detected!\nRemainder: " << removeLeading0s(rem);
    else
        cout << "\nNo error!\nRemainder is 0.";
}

int main()
{
    /* Creating Variables */
    int i;
```

```
    string msg, gen, rem, temp, encoded;

    cout << "\nEnter the frame: ";
    cin >> msg;
    if (!checkInput(msg))
        return 1;
    cout << "Enter the generator: ";
    cin >> gen;
    if (!checkInput(gen))
        return 1;
    if (gen[gen.length() - 1] != '1')
    {
        cout << "\nInput Error!\nLast bit in generator is not 1.\n"
             << endl;
        return 1;
    }
    if (msg.length() < gen.length())
    {
        cout << "\nInput Error!\nLength of frame is smaller than length of
generator.\n"
             << endl;
        return 1;
    }

    /* Cleaning input */
    msg = removeLeading0s(msg);
    gen = removeLeading0s(gen);

    // Appending zero bits to the low-order end of the message

    temp = msg;
    for (i = 0; i < gen.length() - 1; ++i)
        temp += "0";

    // Performing modulo 2 division to find remainder and printing it

    rem = division(temp, gen);
    cout << "\nRemainder: " << removeLeading0s(rem) << endl;

    /* Calculating encoded msg */
    encoded = msg + rem.substr(1);
    cout << "Transmitted frame: " << encoded << endl;

    /* Adding Error */
    char ch;
    cout << "\nWant to add errors ? (Y/N): ";
    cin >> ch;
```

```
    if (ch == 'y' || ch == 'Y')
    {
        int n;
        cout << "\nEnter the index of bit to add the error: ";
        cin >> n;
        encoded[n - 1] = (encoded[n - 1] == '0') ? '1' : '0';
        cout << "\nTransmitted frame with error: " << encoded << endl;

        /* Checking error by CRC */
        CRC(encoded, gen);
    }
    cout << endl;
    return 0;
}
```

Output 1 :

```
Enter the frame: 110101
Enter the generator: 101

Remainder: 11
Transmitted frame: 11010111

Want to add errors ? (Y/N): y

Enter the index of bit to add the error: 2

Transmitted frame with error: 10010111

Applying Cyclic Redundancy Check....

Error detected!
Remainder: 1
```

```
Enter the frame: 1101234

Input Error!
Entered bits contain characters other than 0 or 1.
```

```
Enter the frame: 1101011
Enter the generator: 0000

Input Error!
Entered bits are only zero.
```

```
Enter the frame: 101011
Enter the generator: 110011011

Input Error!
Length of frame is smaller than length of generator.
```

```
Enter the frame: 101011
Enter the generator: 10

Input Error!
Last bit in generator is not 1.
```

Question 2 :

Simulate Hamming code for a given input message.

Solution 2 :

```
/* Created By - ANUJ KUMAR SEN */

#include <iostream>
using namespace std;

void hamming(int *data, int *redundant, int n, int r)
{
    int hammcode[n], count = 0, k = 0, r1 = r;
    for (int i = n; i >= 1; i--)
    {
        if (redundant[r1 - 1] != i)
        {
            hammcode[i] = data[k]; // initializing the hamming code with r bits=0
and data bits as given.
            ++k;
        }
        else
        {
            hammcode[i] = 0;
            --r1;
        }
    }
    for (int i = 0; i < r; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            if ((1 << i) & j) // searching for numbers with (n+1)th bit position=1
            {
                if (hammcode[j] == 1)
                {
                    ++count; // for even parity.
                }
            }
        }
        if (count % 2 == 0) // if bit position=1,update the redundant bit position
according to even parity.
        {
            hammcode[redundant[i]] = 0;
        }
        else
```

```
        {
            hammcode[redundant[i]] = 1;
        }
        count = 0;
    }
    cout << "HAMMING CODE:";
    for (int i = n; i >= 1; i--)
    {
        cout << hammcode[i];
    }
}

int main()
{
    int size, a = 2, r = 1;
    cout << "Enter size of the data ::: ";
    cin >> size;
    while (1)
    {
        if (a >= (size + 1 + r)) // 2^r>=d+r+1
        {
            break;
        }
        ++r;
        a = a * 2;
    }
    int data[size];
    int redundant[5] = {1, 2, 4, 8, 16}, j = r;
    cout << "Enter data ::: " << endl;
    for (int i = 0; i < size; i++)
    {
        cout << "Bit " << i << " : ";
        cin >> data[i];
    }
    if (data[0] == 0)
    {
        cout << "Inappropriate data entered";
        exit(0);
    }
    hamming(data, redundant, size + r, r);
    return 0;
}
```

Output 2 :

```
Enter size of the data ::: 5
Enter data :::
Bit 0 : 1
Bit 1 : 0
Bit 2 : 1
Bit 3 : 1
Bit 4 : 1
HAMMING CODE:110110101
```

```
Enter size of the data ::: 4
Enter data :::
Bit 0 : 1
Bit 1 : 0
Bit 2 : 1
Bit 3 : 0
HAMMING CODE:1010010
```

```
Enter size of the data ::: 5
Enter data :::
Bit 0 : 0
Bit 1 : 1
Bit 2 : 0
Bit 3 : 1
Bit 4 : 1
Inappropriate data entered
```

Question 3 :

Simulate and implement stop and wait protocol for noisy channel.

Solution 3 :

```
/* Created By - ANUJ KUMAR SEN */

#include <iostream>
#include <time.h>
#include <cstdlib>
#include <ctime>
#include <unistd.h>
#include <iomanip>
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()
{
    int frames[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    unsigned long seconds = 5;
    srand(time(NULL));
    timer t;
    cout << "Sender has to send frames : ";
    for (int i = 0; i < 10; 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 != 10);
    return 0;
}
```

Output 3 :

```
Sender has to send frames : 1 2 3 4 5 6 7 8 9 10

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
Acknowledgement : 4
Sending Frame : 5              ---
Timeout
Sending Frame : 5              Received Frame : 5
Acknowledgement : 5
Sending Frame : 6              Received Frame : 6
Delayed Ack
Sending Frame : 6              Received Frame : 6 Duplicate
Acknowledgement : 6
Sending Frame : 7              Received Frame : 7
Acknowledgement : 7
Sending Frame : 8              Received Frame : 8
Acknowledgement : 8
Sending Frame : 9              Received Frame : 9
Delayed Ack
Sending Frame : 9              Received Frame : 9 Duplicate
Acknowledgement : 9
Sending Frame : 10             Received Frame : 10
Acknowledgement : 29261
```

```
Sender has to send frames : 1 2 3 4 5 6 7 8 9 10

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                  ---
Timeout
Sending Frame : 4                  Received Frame : 4
Acknowledgement : 4
Sending Frame : 5                  Received Frame : 5
Acknowledgement : 5
Sending Frame : 6                  Received Frame : 6
Acknowledgement : 6
Sending Frame : 7                  Received Frame : 7
Acknowledgement : 7
Sending Frame : 8                  Received Frame : 8
Acknowledgement : 8
Sending Frame : 9                  Received Frame : 9
Acknowledgement : 9
Sending Frame : 10                 Received Frame : 10
Acknowledgement : 38002
```

Question 4 :

Simulate and implement go back n sliding window protocol.

Solution 4 :

```
/* Created By - ANUJ KUMAR SEN */

#include <bits/stdc++.h>
#include <ctime>
#define ll long long int
using namespace std;

void transmission(ll &i, ll &N, ll &tf, ll &tt)
{
    while (i <= tf)
    {
        int z = 0;
        for (int k = i; k < i + N && k <= tf; k++)
        {
            cout << "Sending Frame " << k << "..." << endl;
            tt++;
        }
        for (int k = i; k < i + N && k <= tf; k++)
        {
            int f = rand() % 2;
            if (!f)
            {
                cout << "Acknowledgment for Frame " << k << "..." << endl;
                z++;
            }
            else
            {
                cout << "Timeout!! Frame Number : " << k << " Not Received" <<
endl;
                cout << "Retransmitting Window..." << endl;
                break;
            }
        }
        cout << "\n";
        i = i + z;
    }
}

int main()
{
```

```
    ll tf, N, tt = 0;
    srand(time(NULL));
    cout << "Enter the Total number of frames : ";
    cin >> tf;
    cout << "Enter the Window Size : ";
    cin >> N;
    ll i = 1;
    transmission(i, N, tf, tt);
    cout << "Total number of frames which were sent and resent are : " << tt <<
endl;
    return 0;
}
```

Output 4 :

```
Enter the Total number of frames : 12
Enter the Window Size : 4
Sending Frame 1...
Sending Frame 2...
Sending Frame 3...
Sending Frame 4...
Acknowledgment for Frame 1...
Timeout!! Frame Number : 2 Not Received
Retransmitting Window...

Sending Frame 2...
Sending Frame 3...
Sending Frame 4...
Sending Frame 5...
Acknowledgment for Frame 2...
Timeout!! Frame Number : 3 Not Received
Retransmitting Window...

Sending Frame 3...
Sending Frame 4...
Sending Frame 5...
Sending Frame 6...
Acknowledgment for Frame 3...
Timeout!! Frame Number : 4 Not Received
Retransmitting Window...

Sending Frame 4...
Sending Frame 5...
Sending Frame 6...
Sending Frame 7...
Acknowledgment for Frame 4...
Acknowledgment for Frame 5...
Timeout!! Frame Number : 6 Not Received
Retransmitting Window...

Sending Frame 6...
Sending Frame 7...
Sending Frame 8...
Sending Frame 9...
Acknowledgment for Frame 6...
Timeout!! Frame Number : 7 Not Received
Retransmitting Window...

Sending Frame 7...
Sending Frame 8...
Sending Frame 9...
Sending Frame 10...
Acknowledgment for Frame 7...
Timeout!! Frame Number : 8 Not Received
Retransmitting Window...

Sending Frame 8...
Sending Frame 9...
Sending Frame 10...
Sending Frame 11...
Timeout!! Frame Number : 8 Not Received
Retransmitting Window...

Sending Frame 8...
Sending Frame 9...
Sending Frame 10...
Sending Frame 11...
Acknowledgment for Frame 8...
Acknowledgment for Frame 9...
Acknowledgment for Frame 10...
Acknowledgment for Frame 11...

Sending Frame 12...
Timeout!! Frame Number : 12 Not Received
Retransmitting Window...

Sending Frame 12...
Timeout!! Frame Number : 12 Not Received
Retransmitting Window...

Sending Frame 12...
Acknowledgment for Frame 12...
```

Question 5 :

Simulate and implement selective repeat sliding window protocol.

Solution 5 :

```
/* Created By - ANUJ KUMAR SEN */

#include <iostream>
using namespace std;
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#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; // tells expected frame
public:
    void input();
    void sender(int);
    void reciever(int);
};

void sel_repeat ::input()
{
    int n; // no of bits for the frame
    int m; // no of frames from n bits

    cout << "Enter the no of bits for the sequence number ::: ";
    cin >> n;

    m = pow(2, n);

    int t = 0;
```

```
    fr_send_at_instance = (m / 2);

    for (int i = 0; i < TOT_FRAMES; i++)
    {
        arr[i] = t;
        t = (t + 1) % m;
    }

    for (int 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";
    }
    reciever(m);
}

void sel_repeat ::reciever(int m)
{
    time_t t;
    int f;
    int f1;
    int a1;
    char ch;
    int i;
    int j;
    srand((unsigned)time(&t));

    for (i = 0; i < fr_send_at_instance; i++)
    {
        if (rcvd_ack[i] == 'n')
        {
            f = rand() % 10;
```

```
            // if = 5 frame is discarded for some reason
            // else frame is correctly recieved

            if (f != 5)
            {

                for (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;
                    }

                if (j == fr_send_at_instance)
                    cout << "RECIEVER : Duplicate frame " << send[i] << "
discarded\n";

                a1 = rand() % 5;

                // if a1 == 3 then ack is lost
                //            else recieved

                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 = 0 then frame damaged
                // else frame lost

                if (ld == 0)
                {
```

```
                    cout << "RECIEVER : Frame " << send[i] << " is damaged\n";
                    cout << "RECIEVER : Negative acknowledgement " << send[i] << "
sent\n";
                }
                else
                {
                    cout << "RECIEVER : 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;
    }

    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()
{
    sel_repeat sr;
    sr.input();
    getch();
    return 0;
}
```

Output 5 :

```
Enter the no of bits for the sequence number ::: 3
 SENDER   : Frame 0 is sent
 SENDER   : Frame 1 is sent
 SENDER   : Frame 2 is sent
 SENDER   : Frame 3 is sent
RECIEVER : Frame 0 recieved correctly
(Acknowledgement 0 recieved)
RECIEVER : Frame 1 recieved correctly
(Acknowledgement 1 recieved)
RECIEVER : Frame 2 recieved correctly
(Acknowledgement 2 lost)
 (SENDER TIMEOUTS --> RESEND THE FRAME)
RECIEVER : Frame 3 recieved correctly
(Acknowledgement 3 recieved)
Do you want to continue (y/n) ::: y

 SENDER   : Frame 4 is sent
 SENDER   : Frame 5 is sent
 SENDER   : Frame 6 is sent
RECIEVER : Frame 4 recieved correctly
(Acknowledgement 4 lost)
 (SENDER TIMEOUTS --> RESEND THE FRAME)
RECIEVER : Frame 5 recieved correctly
(Acknowledgement 5 recieved)
RECIEVER : Frame 6 recieved correctly
(Acknowledgement 6 lost)
 (SENDER TIMEOUTS --> RESEND THE FRAME)
Do you want to continue (y/n) ::: y

 SENDER   : Frame 6 is sent
 SENDER   : Frame 7 is sent
 SENDER   : Frame 0 is sent
RECIEVER : Duplicate frame 6 discarded
(Acknowledgement 6 recieved)
RECIEVER : Frame 7 recieved correctly
(Acknowledgement 7 recieved)
RECIEVER : Frame 0 recieved correctly
(Acknowledgement 0 recieved)
Do you want to continue (y/n) ::: n
```

Question 6 :

Simulate Classful Addressing by taking the IP address (Dotted-Decimal notation) as input and print the corresponding class.

Solution 6 :

```
/* Created By - ANUJ KUMAR SEN */

// 6. Simulate Classful Addressing by taking the IP address (Dotted-Decimal
notation) as input and print the corresponding class.

#include <iostream>
#include <string>
using namespace std;

char findClass(string str)
{
    string arr;
    int i = 0;
    while (str[i] != '.')
    {
        arr = arr + str[i];
        i++;
    }
    int ip = stoi(str);

    if (ip >= 1 && ip <= 126)
    {
        return 'A';
    }
    else if (ip >= 128 && ip <= 191)
    {
        return 'B';
    }
    else if (ip >= 192 && ip <= 223)
    {
        return 'C';
    }
    else if (ip >= 224 && ip <= 239)
    {
        return 'D';
    }
    else if (ip >= 240 && ip <= 255)
    {
```

```
        return 'E';
    }
    else
    {
        return 'I';
    }
}

int main()
{
    string str;
    char ch = 'y';
    do
    {
        cout << "Enter IP : ";
        cin >> str;
        char ipClass = findClass(str);
        if (ipClass == 'I')
        {
            cout << "Invalid IP address" << endl;
        }
        else
        {
            cout << "Given IP address belongs to Class " << ipClass << endl;
        }

        cout << "Do you want to continue(y/n) : ";
        cin >> ch;

        if (ch != 'y')
        {
            break;
        }

    } while (ch == 'y');

    return 0;
}
```

Output 6 :

```
Enter IP : 121.1.15.12
Given IP address belongs to Class A
Do you want to continue(y/n) : y
Enter IP : 181.25.35.10
Given IP address belongs to Class B
Do you want to continue(y/n) : y
Enter IP : 201.2.2.0
Given IP address belongs to Class C
Do you want to continue(y/n) : y
Enter IP : 230.7.4.10
Given IP address belongs to Class D
Do you want to continue(y/n) : y
Enter IP : 251.9.9.9
Given IP address belongs to Class E
Do you want to continue(y/n) : y
Enter IP : 270.5.4.1
Invalid IP address
Do you want to continue(y/n) : n
```

END OF ASSIGNMENT