J.C. Bose University of Science & Technology, YMCA, Faridabad

(A Haryana State Government University)

(Established by Haryana State Legislative Act No. 21 of 2009 & Recognised by UGC Act 1956 u/s 22 to Confer Degrees)

Accredited 'A' Grade by NAAC

Electronics And Computer Engineering

Computer Network Lab

Submitted To:
Mr. Bharat Bhushan

Submitted By:
Manav Bansal
Roll No. 19001015029

INDEX

Experiment 1

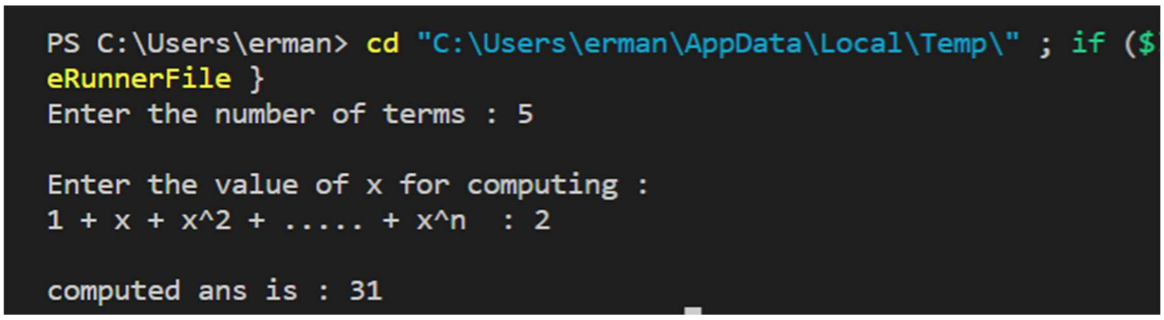
Aim: Implementation of Series

i) $1 + x + x^2 + x^3 + \ldots\ldots\ldots\ldots + x^n$

Code:

```
// 1.i) 1 + x + x^2 + ..... + x^n

#include<iostream>
using namespace std;

int factorial(int n){
    if(n<0){
        cout<<"Enter number should be greater than zero ";
        return -1;
    }
    int fact = 1;
    for(int i=2;i<=n;i++){
        fact = fact*i;
    }
    return fact;
}

int power(int a,int b){
    if(b==0){
        return 1;
    }
    if(b==1){
        return a;
    }
    if(b%2==0){
        return power(a,b/2)*power(a,b/2);
    }
    else{
        return a*power(a,b-1);
    }

}

int main(){

    int n;
    cout<<"Enter the number of terms : ";
    cin>>n;

    int x = 0;
    cout<<"\nEnter the value of x for computing :"<<endl;
    cout<<"1 + x + x^2 + ..... + x^n  : ";
    cin>>x;

    float ans = 1;
    for(int i=1; i<n; i++){
        int nthTerm = power(x,i);
        ans += nthTerm;
    }

    cout<<"\ncomputed ans is : "<<ans;
    return 0;
}
```

Output-



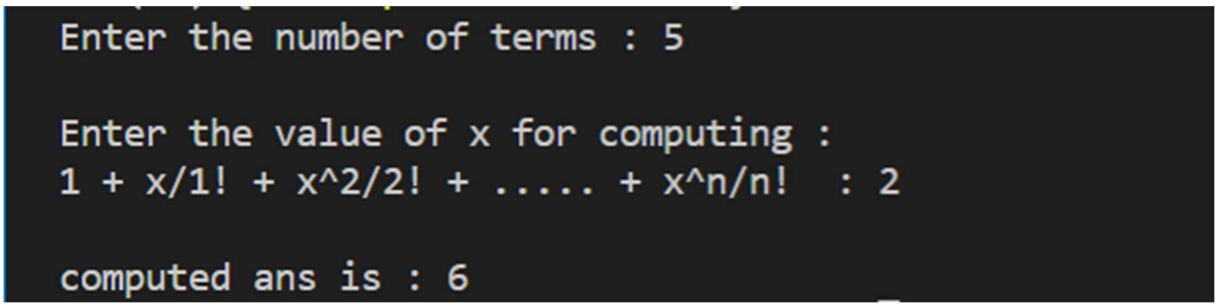
```
PS C:\Users\erman> cd "C:\Users\erman\AppData\Local\Temp\" ; if ($
eRunnerFile }
Enter the number of terms : 5

Enter the value of x for computing :
1 + x + x^2 + ..... + x^n  : 2

computed ans is : 31
```

ii) 1+ x/1! + x^2/2! + x^3/3! + …………+ x^n /n!

Code:

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

int factorial(int n){
    if(n<0){
        cout<<"Enter number should be greater than zero ";
        return -1;
    }
    int fact = 1;
    for(int i=2;i<=n;i++){
        fact = fact*i;
    }
    return fact;
}

int power(int a,int b){
    if(b==0){
        return 1;
    }
    if(b==1){
        return a;
    }
    if(b%2==0){
        return power(a,b/2)*power(a,b/2);
    }
    else{
        return a*power(a,b-1);
    }

}

int main(){

    int n;
    cout<<"Enter the number of terms : ";
    cin>>n;

    int x = 0;
    cout<<"\nEnter the value of x for computing :"<<endl;
    cout<<"1 + x/1! + x^2/2! + ..... + x^n/n!  : ";
    cin>>x;

    float ans = 1.0;
    for(int i=1; i<n; i++){
        float nthTerm = power(x,i)/factorial(i);
        ans += nthTerm;
    }

    cout<<"\ncomputed ans is : "<<ans;
    return 0;
}
```

Output:



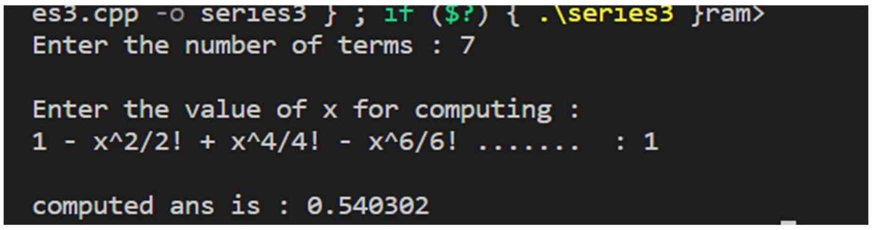
```
Enter the number of terms : 5

Enter the value of x for computing :
1 + x/1! + x^2/2! + ..... + x^n/n!  : 2

computed ans is : 6
```

iii) 1 - $x^2/2!$ + $x^4/4!$ - $x^6/6!$ ………….+ $(-1)^n x^{2n}/(2n)!$

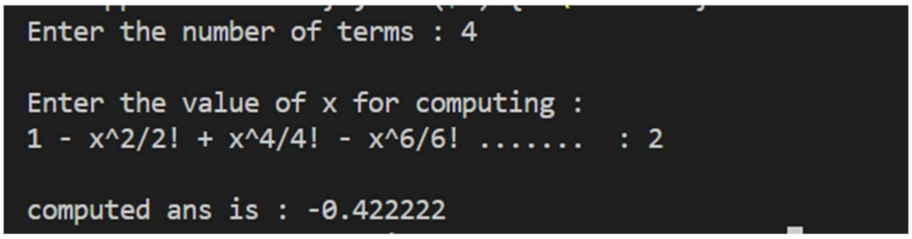
Code:

```
//1 iii) 1 - x^2/2! + x^4/4! - x^6/6! ………….+ (-1)^n * x^2n/(2n)!

#include<iostream>
using namespace std;

float factorial(int n){
    if(n<0){
        cout<<"Enter number should be greater than zero ";
        return -1;
    }
    float fact = 1;
    for(int i=2;i<=n;i++){
        fact = fact*i;
    }
    return fact;
}

int power(int a,int b){
    if(b==0){
        return 1;
    }
    if(b==1){
        return a;
    }
    if(b%2==0){
        return power(a,b/2)*power(a,b/2);
    }
    else{
        return a*power(a,b-1);
    }

}

int main(){

    int n;
    cout<<"Enter the number of terms : ";
    cin>>n;

    int x = 0;
    cout<<"\nEnter the value of x for computing :"<<endl;
    cout<<"1 - x^2/2! + x^4/4! - x^6/6! .......  : ";
    cin>>x;
    float ans = 1.0;
    for(int i=1; i<n; i++){
        float nthTerm = (power(-1,i) * power(x,2*n))/factorial(2*i);
        ans += nthTerm;
    }

    cout<<"\ncomputed ans is : "<<ans;
    return 0;
}
```

Output:



```
es3.cpp -o series3 } ; if ($?) { .\series3 }ram>
Enter the number of terms : 7

Enter the value of x for computing :
1 - x^2/2! + x^4/4! - x^6/6! .......  : 1

computed ans is : 0.540302
```



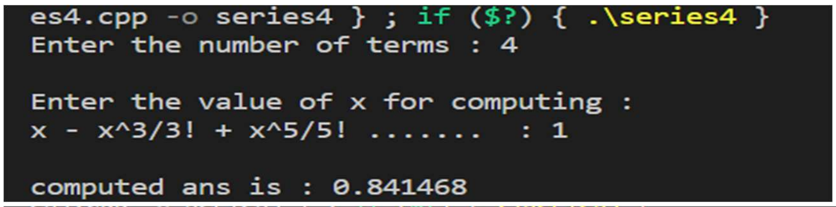
```
Enter the number of terms : 4

Enter the value of x for computing :
1 - x^2/2! + x^4/4! - x^6/6! .......  : 2

computed ans is : -0.422222
```

iv) x - $x^3/3!$ + $x^5/5!$ - ………….+ $(-1)^{n+1}x^{2n+1}/(2n+1)!$

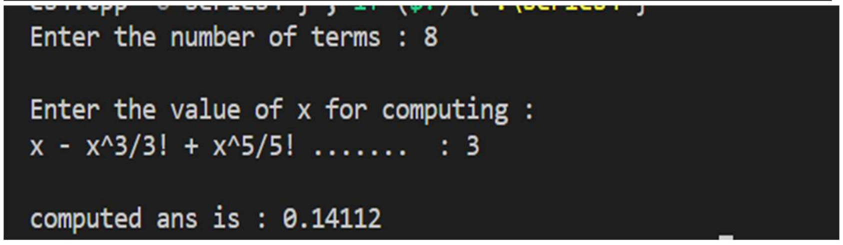
Code:

```
//1 iv) x - x^3/3! + x^5/5! - ………….

#include<iostream>
using namespace std;

float factorial(int n){
    if(n<0){
        cout<<"Enter number should be greater than zero ";
        return -1;
    }
    float fact = 1;
    for(int i=2;i<=n;i++){
        fact = fact*i;
    }
    return fact;
}

int power(int a,int b){
    if(b==0){
        return 1;
    }
    if(b==1){
        return a;
    }
    if(b%2==0){
        return power(a,b/2)*power(a,b/2);
    }
    else{
        return a*power(a,b-1);
    }
}

int main(){
    int n;
    cout<<"Enter the number of terms : ";
    cin>>n;

    int x = 0;
    cout<<"\nEnter the value of x for computing :"<<endl;
    cout<<"x - x^3/3! + x^5/5! .......  : ";
    cin>>x;
    float ans = 0;
    for(int i=0; i<n; i++){
        float nthTerm = (power(-1,i) *
power(x,2*i+1))/factorial(2*i+1);
        ans += nthTerm;
    }
    cout<<"\ncomputed ans is : "<<ans;
    return 0;
}
```

Output:



```
es4.cpp -o series4 } ; if ($?) { .\series4 }
Enter the number of terms : 4

Enter the value of x for computing :
x - x^3/3! + x^5/5! .......  : 1

computed ans is : 0.841468
```



```
Enter the number of terms : 8

Enter the value of x for computing :
x - x^3/3! + x^5/5! .......  : 3

computed ans is : 0.14112
```

Result: Implemented the series using C++ Language

Experiment 2

Aim: Write a program to implement Diffie Hellman Algorithm

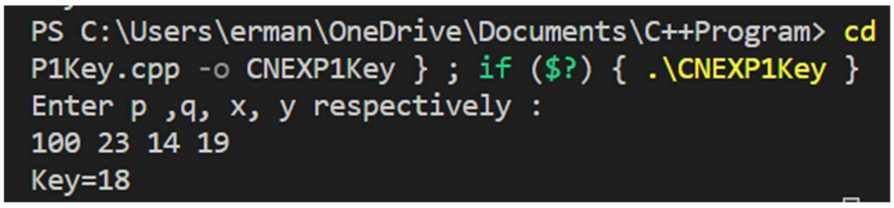
The Diffie–Hellman (DH) Algorithm is a key-exchange protocol that enables two parties communicating over public channel to establish a mutual secret without it being transmitted over the Internet. DH enables the two to use a public key to encrypt and decrypt their conversation or data using symmetric cryptography.

Code:

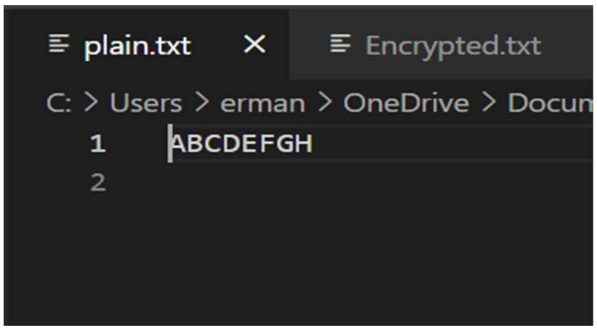
```
#include<iostream>
#include<fstream>
using namespace std;
#define ll long long

ll int power(int a,int b){
    if(b==0 or b==1) return 1;
    if (b&1==1)
        return (a*power(a*a,b/2));
    else
        return power(a*a,b/2);

}
long long int powInt(long long int p, long long int x, long long int
q) {
    if (p == 0)
        return 0;
    if (x == 0)
        return 1;
    long long int y;
    if (x % 2 == 0) {
        y = powInt(p, x / 2, q);
        y = (y * y) % q;
    }
    else {
        y = p % q;
        y = (y * powInt(p, x - 1, q) % q) % q;
    }

    return ((y + q) % q);
}
```

```
long long int powerStrings(const string& sa, const string& sb, long
long int q) {
    long long int a = 0, b = 0;
    for (char i : sa)
        a = (a * 10 + (i - '0')) % q;
    for (char i : sb)
        b = (b * 10 + (i - '0')) % (q - 1);
    return powInt(a, b, q);
}

class Key{
    private:
        ll int k;
    public:
        Key(ll int p,ll int q, const string& x,const string& y){
            k=powerStrings(to_string(p),x,q);
            k=powerStrings(to_string(k),y,q);
            cout<<"Key="<<k;
        }
        int returnKey() const {
        return (int)k;
    }

};

int main(){

    int p,q,x,y;
    cout<<"Enter p ,q, x, y respectively :\n";
    cin>>p>>q>>x>>y;

    Key key(p,q,to_string(x),to_string(y));
    ifstream in;
    ofstream out;
    in.open("./plain.txt", ios::in);
    out.open("./encrypted.txt", ios::out);

    string str;
    in>>str;
    string res;
    for(char i : str) {
        res += (char)(i + key.returnKey());
    }
    out<<res;
    in.close();
    out.close();
    return 0;
}
```

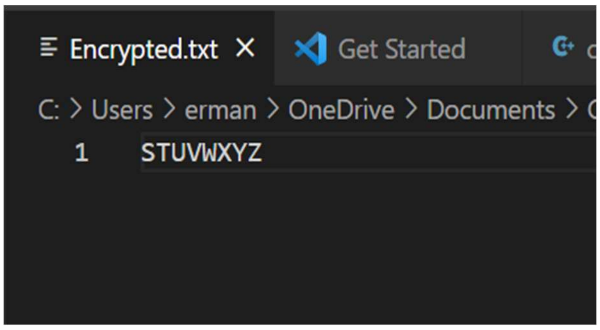
Output:



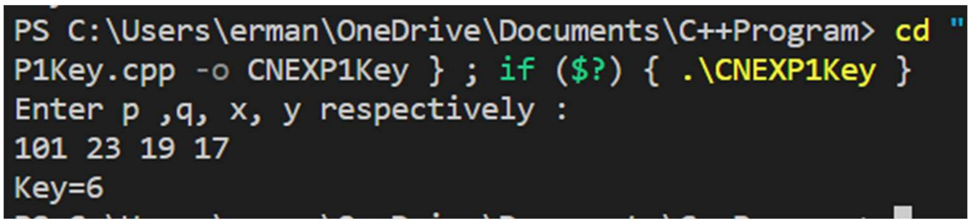
```
PS C:\Users\erman\OneDrive\Documents\C++Program> cd
P1Key.cpp -o CNEXP1Key } ; if ($?) { .\CNEXP1Key }
Enter p ,q, x, y respectively :
100 23 14 19
Key=18
```



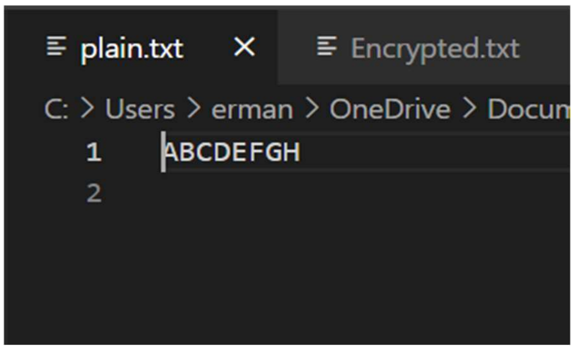
```
plain.txt × | Encrypted.txt
C: > Users > erman > OneDrive > Docun
1  ABCDEFGH
2
```



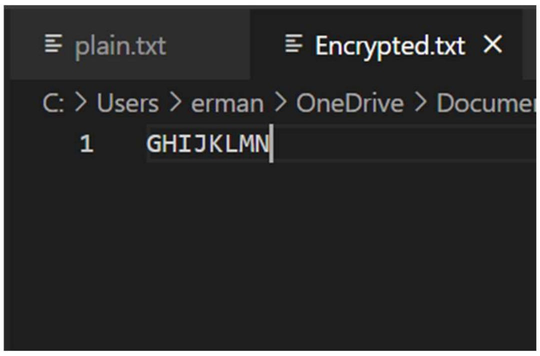
```
Encrypted.txt × | Get Started
C: > Users > erman > OneDrive > Documents >
1  STUVWXYZ
```



```
PS C:\Users\erman\OneDrive\Documents\C++Program> cd "
P1Key.cpp -o CNEXP1Key } ; if ($?) { .\CNEXP1Key }
Enter p ,q, x, y respectively :
101 23 19 17
Key=6
```



```
plain.txt × | Encrypted.txt
C: > Users > erman > OneDrive > Docun
1  ABCDEFGH
2
```



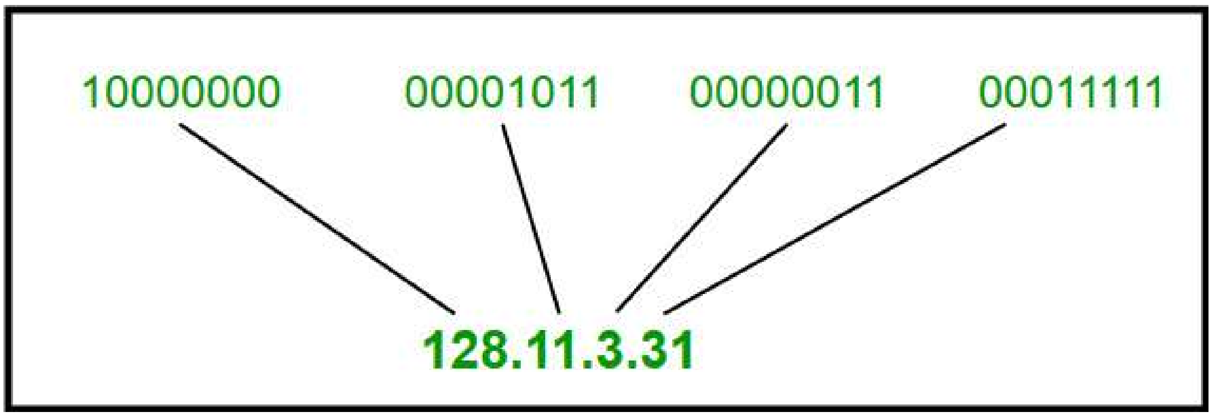
```
plain.txt | Encrypted.txt ×
C: > Users > erman > OneDrive > Documer
1  GHIJKLMN
```

Result: Implemented Diffie Hellman Algorithm

Experiment 3

Aim: Write a program to implement IP Addressing

IP address is an address having information about how to reach a specific host, especially outside the LAN. An IP address is a 32 bit unique address having an address space of 232.
Generally, there are two notations in which IP address is written, dotted decimal notation and hexadecimal notation.
Dotted Decimal Notation:

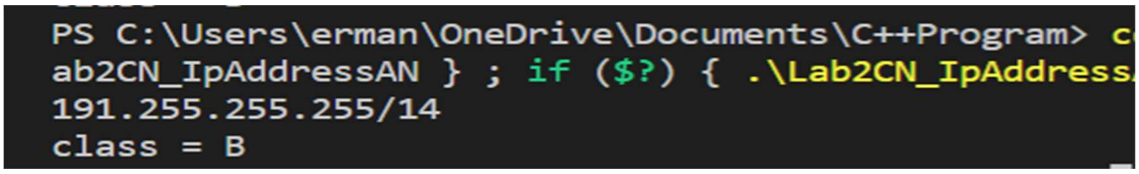


Code:

```
#include <iostream>
#include <string>

using namespace std;
int dec(string &s) {
    int res = 0;
    int two = 1;
    for (int i = (int)s.length() - 1; i >= 0; i--) {
        if (s[i] == '1')
            res += two;
        two *= 2;
    }
    return res;
}
bool starts_with(string s,string pattern){
    return s.substr(0,pattern.length()) == pattern;
}
int main() {
    string s = "10111111111111111111111111111111";
    string o[4];
    o[0] = s.substr(0, 8);
    o[1] = s.substr(8, 8);
    o[2] = s.substr(16, 8);
    o[3] = s.substr(24, string::npos);
    cout << dec(o[0]) << '.';
    cout << dec(o[1]) << '.';
    cout << dec(o[2]) << '.';
    cout << dec(o[3]) << '/';

    if (starts_with(s,"0")) {
        cout << 7 << '\n';
        cout << "class = " << 'A';
    } else if (starts_with(s,"10")) {
        cout << 14 << '\n';
        cout << "class = " << 'B';
    } else if (starts_with(s,"110")) {
        cout << 21 << '\n';
        cout << "class = " << 'C';
    } else if (starts_with(s,"1110")) {
        cout << 28 << '\n';
        cout << "class = " << 'D';
    } else {
        cout << 28 << '\n';
        cout << "class = " << 'E';
    }
    return 0;
}
```

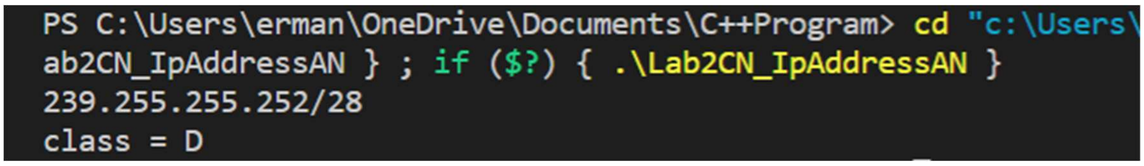
Output:

For I/P 10111111111111111111111111111111



```
PS C:\Users\erman\OneDrive\Documents\C++Program> c
ab2CN_IpAddressAN } ; if ($?) { .\Lab2CN_IpAddress
191.255.255.255/14
class = B
```

For I/P 11101111111111111111111111111100



```
PS C:\Users\erman\OneDrive\Documents\C++Program> cd "c:\Users\
ab2CN_IpAddressAN } ; if ($?) { .\Lab2CN_IpAddressAN }
239.255.255.252/28
class = D
```

Result: Implemented IP Addressing uing C++ programming language

Experiment 4

Aim: Write a program to implement LRC, VRC, Block Parity

1. Vertical Redundancy Check (VRC) **:**

Vertical Redundancy Check is the error detection method which is used by upper layers to detect error in data. The other name for VRC is **Parity Check**. A redundant bit which is also named as parity bit is added to each data unit. This method includes even parity and odd parity. When the total number of 1s in data is even that there is **even parity** and when the total number of 1s in data is to be odd that indicates there is **odd parity** in data.

2. Longitudinal Redundancy Check (LRC) **:**

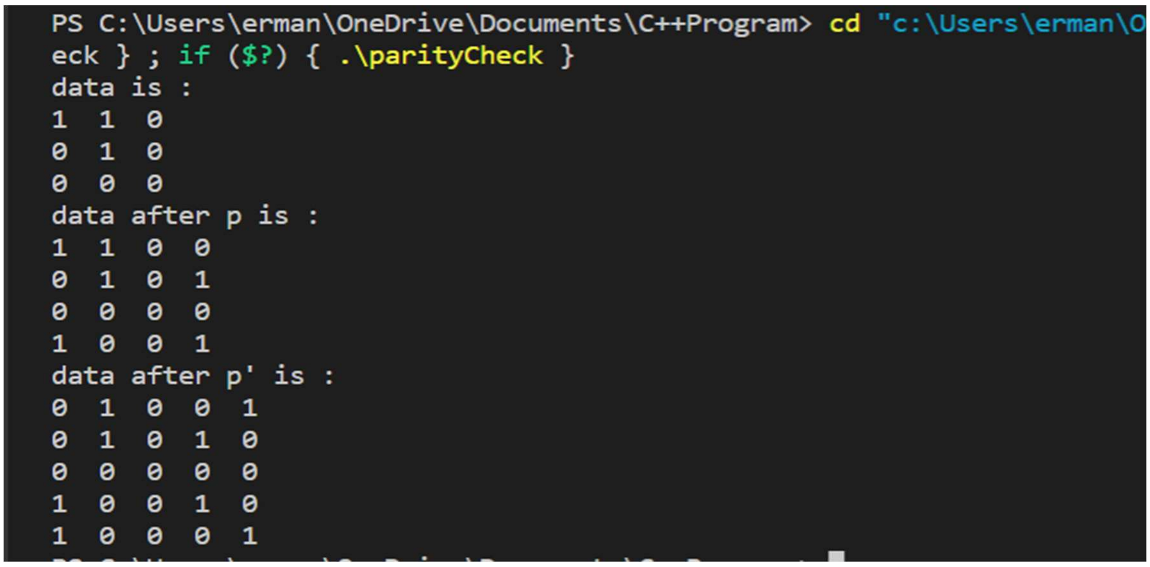
Longitudinal Redundancy Check (LRC) is the error detection method which is used by upper layers to detect error in data. The other name for VRC is **2-D parity check**. In this method, data which the user want to send is organized into tables of rows and columns. To detect an error, a **redundant bit** is added to the whole block after addition this block is transmitted to receiver side. This redundant bit is used by receiver to detect error. If there is no error, receiver accepts the data and discards the redundant row of bits.

Code:

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

void print(int arr[][5],int m,int n){
    for (int i=0;i<m;i++){
        for (int j=0;j<n;j++){
            cout<<arr[i][j]<<"  ";
        }
        cout<<endl;
    }
}

int xor1(int a,int b){
    if(a==b) return 0;
    else return 1;
}

int main(){

    int data[5][5]={0};
    int p;
```

```
for (int i=0;i<3;i++){
    for (int j=0;j<3;j++){
        data[i][j]=rand()%2;
    }
}

cout<<"data is :"<<endl;
print(data,3,3);

//4th column row
for (int i=0;i<3;i++){
    p=0;
    for (int j=0;j<3;j++){
        p = xor1(data[i][j],p);
    }
    data[i][3]=p;
}

for (int i=0;i<3;i++){
    p=0;
    for (int j=0;j<3;j++){
         p = xor1(data[j][i],p);
    }
    data[3][i]=p;
}
p=0;
for (int i=0;i<3;i++){

    for (int j=0;j<3;j++){
         p = xor1(data[i][j],p);
    }
}
data[3][3]=p;

cout<<"data after p is :"<<endl;
print(data,4,4);

data[0][0] = 1 - data[0][0];


// 5th column

 for (int i=0;i<4;i++){
    p=0;
    for (int j=0;j<4;j++){
        p = xor1(data[i][j],p);
    }
    data[i][4]=p;
    }

    for (int i=0;i<4;i++){
        p=0;
        for (int j=0;j<4;j++){
             p = xor1(data[j][i],p);
        }
        data[4][i]=p;
    }

    p=0;
    for (int i=0;i<4;i++){
        for (int j=0;j<4;j++){
             p = xor1(data[i][j],p);
        }
    }
    data[4][4]=p;

    cout<<"data after p' is :"<<endl;
    print(data,5,5);

return 0;
}
```

Output:



```
PS C:\Users\erman\OneDrive\Documents\C++Program> cd "c:\Users\erman\O
eck } ; if ($?) { .\parityCheck }
data is :
1  1  0
0  1  0
0  0  0
data after p is :
1  1  0  0
0  1  0  1
0  0  0  0
1  0  0  1
data after p' is :
0  1  0  0  1
0  1  0  1  0
0  0  0  0  0
1  0  0  1  0
1  0  0  0  1
```

Result: Implemented program of LRC, VRC, Block Parity

Experiment 5

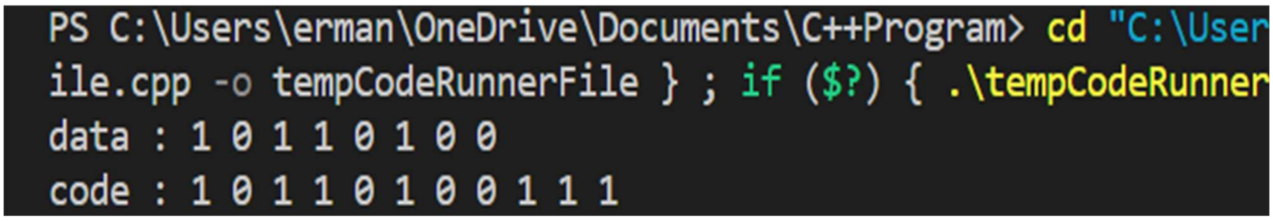
Aim: Write a program to implement CRC

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to digital data.

Code:

```
#include<bits/stdc++.h>
using namespace std;

int xor1(int a, int b){
    if(a==b) return 0;
    else return 1;
}
void print(vector<int> a,int x,int y){
    for(int i=x; i<=y;i++){
        cout<<a[i]<<" ";
    }
}

int main(){

    vector<int> data = {1,0,1,1,0,1,0,0};
    vector<int> code(data.size()+3,0);

    for(int i=0;i<data.size();i++){
        code[i] = data[i];
    }
    vector<int> div = {1,1,0,1};
    for(int i=0;i<8;i++){
        int m = code[i];
        for(int j=0;j<4;j++){
            code[i+j] = xor1(data[i+j],div[j]*m);
        }

    }
    cout<<"data : ";
    print(data,0,7);
    cout<<endl<<"code : ";
     print(data,0,7);
     print(code,8,10);


    return 0;
}
```

Output:



```
PS C:\Users\erman\OneDrive\Documents\C++Program> cd "C:\User
ile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunner
data : 1 0 1 1 0 1 0 0
code : 1 0 1 1 0 1 0 0 1 1 1
```

Result: Implemented program of CRC

Experiment 6

Aim: Write a program to implement Hamming Code

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver. It is technique developed by R.W. Hamming for error correction.
Redundant bits –
Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer.
The number of redundant bits can be calculated using the following formula:

$2^r \geq m + r + 1$

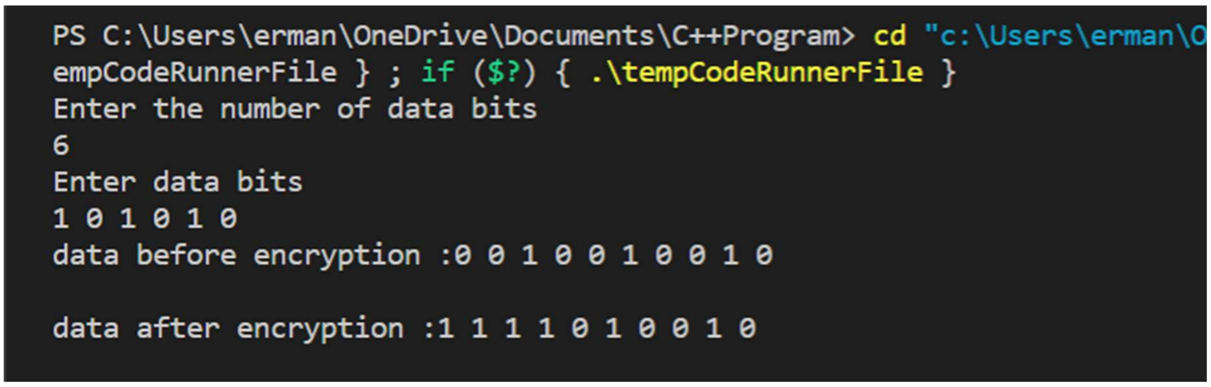
where, r = redundant bit, m = data bit

Suppose the number of data bits is 7, then the number of redundant bits can be calculated using:
$= 2^4 \geq 7 + 4 + 1$
Thus, the number of redundant bits= 4

Code:

```
#include<bits/stdc++.h>
using namespace std;

bool isPowerOfTwo (int x)
{
    return x && (!(x&(x-1)));
}

void print(vector<int>& arr){

    for(int i=1;i<arr.size();i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}

void dec2bin(int n,vector<int>& b,int k){
    for(int i=0;i<k;i++){
        b[i]=n%2;
        n/=2;
    }
    reverse(b.begin(),b.end());
}


int xor1(int a,int b){
    if(a==b) return 0;
    else return 1;
}

void hamming(vector<int>& code,int n,int k){
    int maxBits = (int)((log(n + k) /log(2)) + 1) + 1;
    vector<int> b(maxBits,0);
    int p;
    for(int i=0;i<k;i++){
        int pos = 1<<i;
        p=0;

        for(int j=pos;j<=n+k;j++){
            dec2bin(j,b,k);
            // cout<<"b for "<<k<<" is ";
            // print(b);
            // cout<<endl<<endl;
            if(b[i]==1){
                p=xor1(p,code[j]);
            }
        }
        code[pos]=p;
    }
    print(code);
    cout<<endl;
}

int main(){

    int n;
    cout<<"Enter the number of data bits "<<endl;
    cin>>n;
    vector<int> data(n,0);
    cout<<"Enter data bits "<<endl;
    for(int i=0;i<n;i++){
        cin>>data[i];
    }
    int k=1;

    while((1<<k) < n+k+1){
        k++;
    }
    vector<int> code(n+k+1,0);
    int ndx=0;  //index for data bit
    for(int i=1;i<=n+k;i++){
        if(isPowerOfTwo(i)){
            code[i]=0;
        }
        else{
            code[i]=data[ndx++];
        }
    }

    cout<<"data before encryption :";
    print(code);
    cout<<endl;

    cout<<"data after encryption :";
    hamming(code,n,k);

    return 0;
}
```

Output:



```
PS C:\Users\erman\OneDrive\Documents\C++Program> cd "c:\Users\erman\O
empCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Enter the number of data bits
6
Enter data bits
1 0 1 0 1 0
data before encryption :0 0 1 0 0 1 0 0 1 0

data after encryption :1 1 1 1 0 1 0 0 1 0
```

Result: Implemented program of Hamming Code

Experiment 7

Aim: Write a program to implement Slotted Aloha

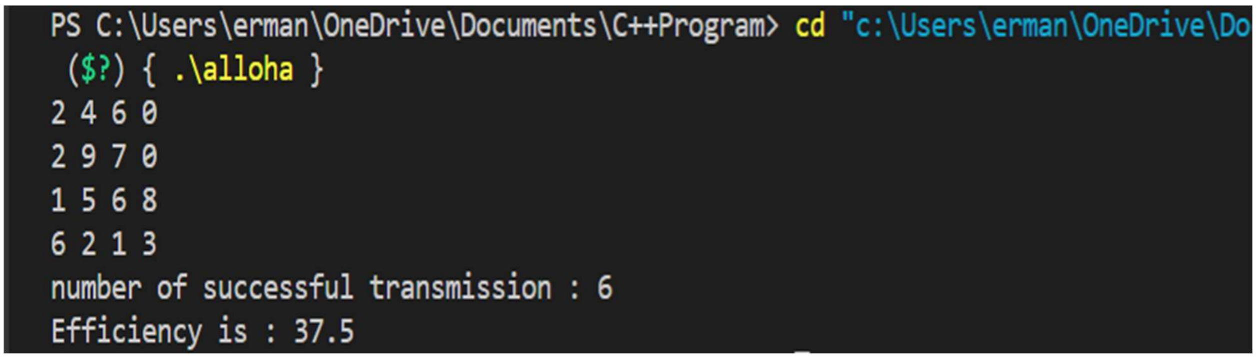
ALOHA is a medium access control (MAC) protocol for transmission of data via ashared network channel. Using this protocol, several data streams originating from multiple nodes are transferred through a multi-point transmission channel. There are two types of ALOHA protocols – Pure ALOHA and Slotted ALOHA.

Pure aloha is used whenever data is available for sending over a channel at stations, whereas slotted aloha is designed to overcome the problem of pure aloha because there is a high possibility of frame hitting in pure aloha

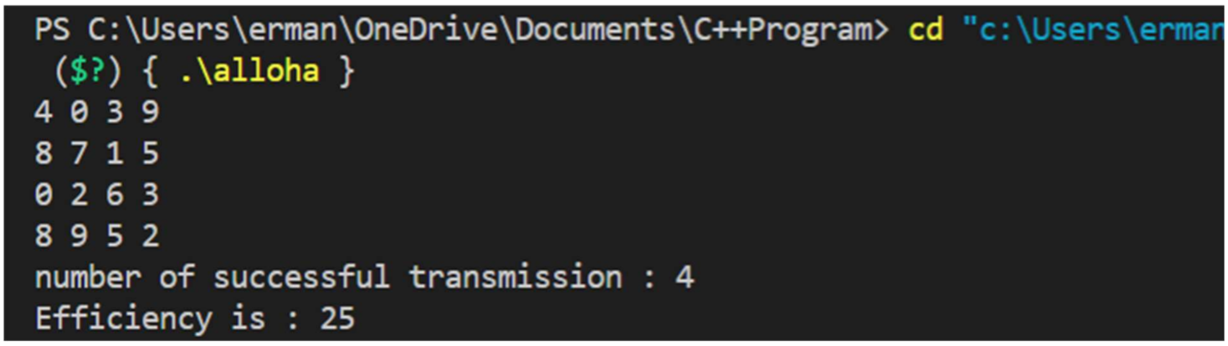
Code:

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n=4,m=4;
    vector<vector<int>> transmitter( n , vector<int> (m, 0));
    int t=0;
    srand(time(0));      //unique random number generator
    for(int i=0;i<transmitter.size();i++){
        vector<bool> check(10,false);
        for(int j=0;j<transmitter[0].size();j++){
            int number = rand()%10;
            //checking that number is not getting repeat
            while(check[number]) {
                number = rand()%10;
            }
            check[number] = true;
            transmitter[i][j] = number;
        }
    }
    for(int i=0;i<transmitter.size();i++){
        for(int j=0;j<transmitter[0].size();j++){
            cout<<transmitter[i][j]<<" ";
        }
        cout<<endl;
    }

    vector<int> count(10,0);
    for(int i=0;i<transmitter.size();i++){
        for(int j=0;j<transmitter[0].size();j++){
            count[transmitter[i][j]]++;
        }
    }
    int ans=0;
    for(int i:count){
        if(i==1){
            ans++;
        }
    }

    cout<<"number of successful transmission : "<<ans<<endl;
    cout<<"Efficiency is : "<<((float)ans/(n*m)) * 100;
    return 0;
}
```

Output:



```
PS C:\Users\erman\OneDrive\Documents\C++Program> cd "c:\Users\erman\OneDrive\Do
 ($?) { .\alloha }
2 4 6 0
2 9 7 0
1 5 6 8
6 2 1 3
number of successful transmission : 6
Efficiency is : 37.5
```



```
PS C:\Users\erman\OneDrive\Documents\C++Program> cd "c:\Users\erman
 ($?) { .\alloha }
4 0 3 9
8 7 1 5
0 2 6 3
8 9 5 2
number of successful transmission : 4
Efficiency is : 25
```

Result: Implemented program of Slotted Aloha and its efficiency

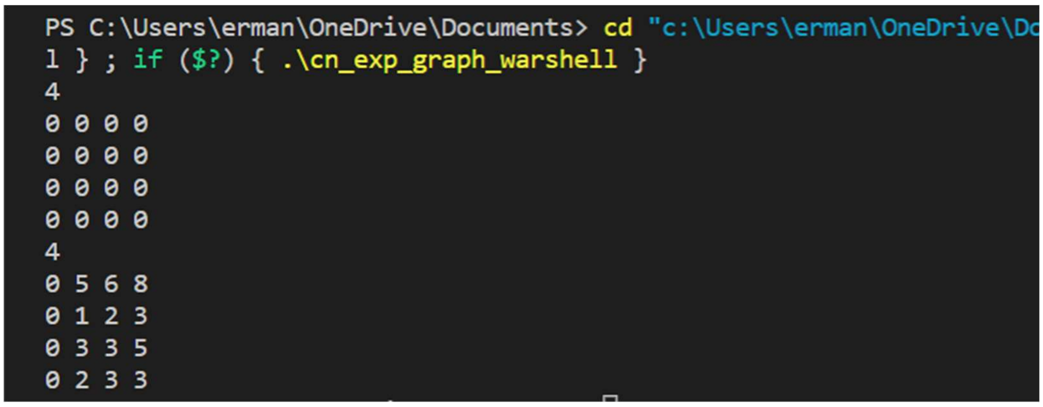
Experiment 8

Aim: Write a program to find number of paths for each pairs of vertices (u,v) in a directed graph using matrix exponentiation and matrix addition and Floyds-Warshall Algo

Code:

```
#include <bits/stdc++.h>
using namespace std;

void matmul(int a[][4], int b[][4], int c[][4]);
void matadd(int a[][4], int b[][4], int c[][4]);
void matdisp(int c[][4]);

int main()
{
    int b[4][4], c[4][4];
    int a[4][4] = {0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0};
    int I[4][4] = {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1};
    int i, j, k;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            c[i][j] = 0;

    cout << i << endl;
    matdisp(c);
    for (i = 0; i < 4; i++)
    {
        matmul(a, I, I);
        matadd(c, I, c);
    }
    cout << i << endl;
    matdisp(c);
    }

    void matmul(int a[][4], int b[][4], int c[][4]){
    int i, j, k, s;
    int t[4][4];
    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < 4; j++)
        {
            s = 0;
            for (k = 0; k < 4; k++)
                s += a[i][k] * b[k][j];
                t[i][j] = s;
        }
    }
        for (i = 0; i < 4; i++)
        {
            for (j = 0; j < 4; j++)
            c[i][j] = t[i][j];
        }
    }

    void matadd(int a[][4], int b[][4], int c[][4])
    {
    int t[4][4];
    int i, j;
    for (i = 0; i < 4; i++)
        {
        for (j = 0; j < 4; j++)
            t[i][j] = a[i][j] + b[i][j];
        }
        for (i = 0; i < 4; i++)
        {
            for (j = 0; j < 4; j++)
                c[i][j] = t[i][j];
        }
    }

    void matdisp(int c[][4])
    {
    int i, j;
    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < 4; j++)
        cout << c[i][j]<<" ";
        cout << endl;
    }
}
```

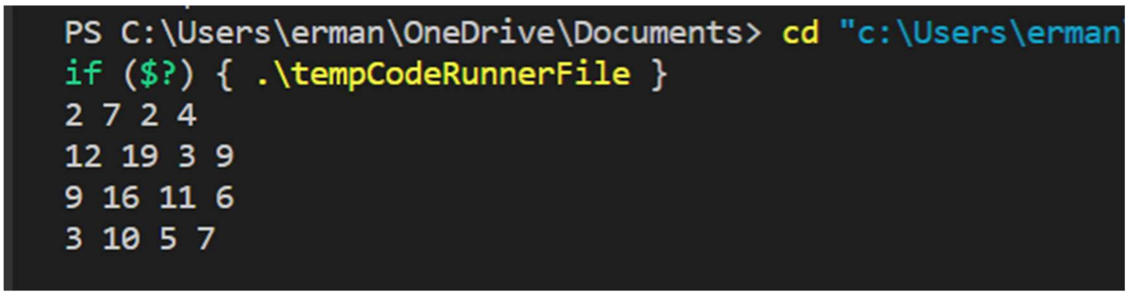
Output:



```
PS C:\Users\erman\OneDrive\Documents> cd "c:\Users\erman\OneDrive\Dc
l } ; if ($?) { .\cn_exp_graph_warshell }
4
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
4
0 5 6 8
0 1 2 3
0 3 3 5
0 2 3 3
```

Code:

```
Floyd-Warshall Algorithm

#include <bits/stdc++.h>
using namespace std;
int main()
{
    long long int p[4][4] = {{2, 7, 2, 4}, {INT_MAX, INT_MAX, 3,
INT_MAX},
    {INT_MAX, INT_MAX, INT_MAX, 6}, {3, INT_MAX, INT_MAX, INT_MAX}};
    for (int k = 0; k < 4; k++)
    {
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            p[i][j] = min(p[i][j], p[i][k] + p[k][j]);
    }
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            cout << p[i][j] << " ";
        }
        cout << endl;
        }
    cout << endl;
}


Output:
```



```
PS C:\Users\erman\OneDrive\Documents> cd "c:\Users\erman
if ($?) { .\tempCodeRunnerFile }
2 7 2 4
12 19 3 9
9 16 11 6
3 10 5 7
```

Result: Program implemented to find number of paths for each pairs of vertices (u,v) in a directed graph using matrix exponentiation and matrix addition and Floyds-Warshall Algo

Experiment 9

Aim: Write a program to implement byte Stuffing and DeStuffing

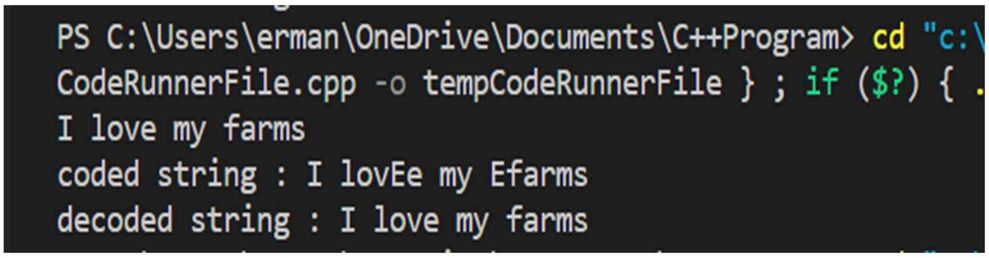
Byte stuffing is a mechanism to convert a message formed of a sequence of bytes that may contain reserved values such as frame delimiter, into another byte sequence that does not contain the reserved values.

Code:

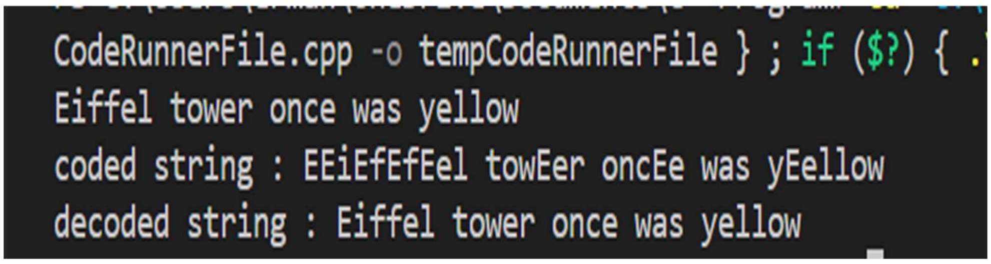
```
#include<iostream>
#include<vector>
#include<string.h>

using namespace std;

void stuffing(string d,string &code){

    int n = d.length();

    for(int i=0;i<n;i++){
        if(toupper(d[i])=='E' || toupper(d[i])=='F'){
            code.push_back('E');
        }
        code.push_back(d[i]);
    }

}

void deStuffing(string code,string &deCode){

    int n = code.length();

    for(int i=0;i<n;i++){
        if(code[i]=='E'){
            if(code[i+1]=='E'){
               deCode.push_back(code[i]);
            }
            else{
                continue;
            }
        }
```

```
            else
            deCode.push_back(code[i]);
        }

    }

    int main(){

        string d;
        getline(cin,d);
        string code = "";
        string deCode = "";
        stuffing(d,code);
        deStuffing(code,deCode);

        cout<<"coded string : "<<code<<endl;

        cout<<"decoded string : "<<deCode;

        return 0;
    }
```

Output:



```
PS C:\Users\erman\OneDrive\Documents\C++Program> cd "c:\
CodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .
I love my farms
coded string : I lovEe my Efarms
decoded string : I love my farms
```



```
CodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .
Eiffel tower once was yellow
coded string : EEiEfEfEel towEer oncEe was yEellow
decoded string : Eiffel tower once was yellow
```

Result : Implemented program of **byte** Stuffing and DeStuffing

Experiment 10

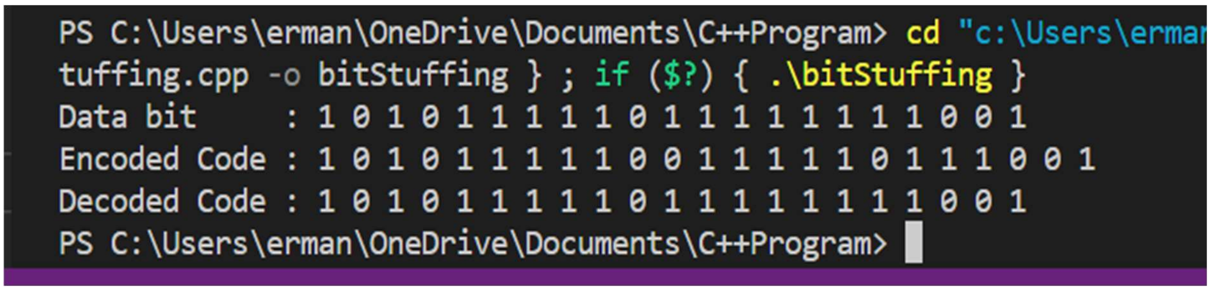
Aim: Write a program to implement bit Stuffing and DeStuffing

Bit stuffing refers to the insertion of one or more bits into a data transmission as a way to provide signaling information to a receiver. The receiver knows how to detect, remove or disregard the stuffed bits.

Code:

```
#include<iostream>
#include<vector>
#include<string.h>

using namespace std;

void print(string s){
    int n = s.length();
    int i=0;
    while(i<n){
        cout<<s[i++]<<" ";
    }
    cout<<endl;
    return;
}
bool isMatching(string data,string pattern){

    if(data.size()<pattern.size()){
        return false;
    }
    for(int i=0;i<pattern.size();i++){
        if(data[i]!=pattern[i]){
            return false;
        }
    }
    return true;

}
void bitStuffing(string &data, string &code,string &pattern){

    int i=0;
    while(i<data.length()){
        if(isMatching(data.substr(i),pattern)){
            int j=0;
            while(j<pattern.length()){
                code+=data[i++];
                j++;
            }
            code+='0';
        }
        else{
            code.push_back(data[i++]);
        }
    }

}

void bitDeStuffing(string &data, string &code,string &pattern){

    int i=0;
    while(i<data.length()){
        if(isMatching(data.substr(i),pattern)){
            int j=0;
            while(j<pattern.length()){
                code+=data[i++];
                j++;
            }
            i++;
        }
        else{
            code.push_back(data[i++]);
        }
    }

}

int main(){

    string data = "101011111011111111001";
    string code = "";
    string pattern = "011111";
    string deCoded = "";
    string dePattern = "011111";

    bitStuffing(data, code, pattern);
    cout<<"Data bit     : ";
    print(data);
    cout<<"Encoded Code : ";
    print(code);
    bitDeStuffing(code,deCoded, dePattern);
    cout<<"Decoded Code : ";
    print(deCoded);

    return 0;
}
```

Output:



```
PS C:\Users\erman\OneDrive\Documents\C++Program> cd "c:\Users\erman
tuffing.cpp -o bitStuffing } ; if ($?) { .\bitStuffing }
Data bit     : 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1
Encoded Code : 1 0 1 0 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1
Decoded Code : 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1
PS C:\Users\erman\OneDrive\Documents\C++Program>
```

Result: Implemented the program of bit Stuffing and DeStuffing

Experiment 11

Aim: Identification of network topology

The arrangement of a network that comprises nodes and connecting lines via sender and receiver is referred to as network topology. The various network topologies are:

Mesh Topology:

In a mesh topology, every device is connected to another device via a particular channel. In Mesh Topology, the protocols used are AHCP (Ad Hoc Configuration Protocols), DHCP (Dynamic Host Configuration Protocol), etc.

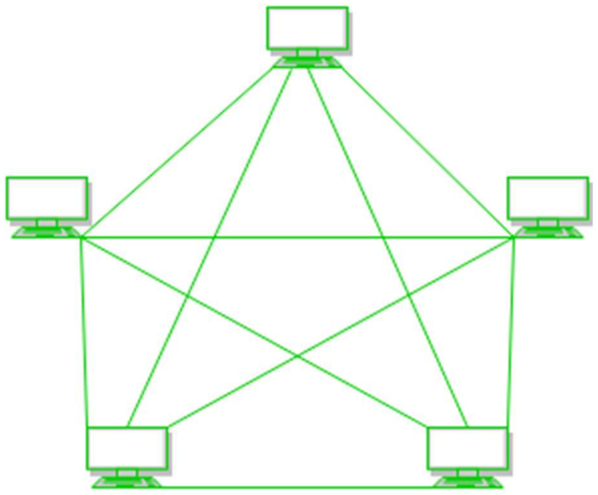


Figure 1: Every device is connected with another via dedicated channels. These channels are known as links.

- Suppose, the N number of devices are connected with each other in a mesh topology, the total number of ports that are required by each device is N-1. In Figure 1, there are 5 devices connected to each other, hence the total number of ports required by each device is 4. Total number of ports required=N*(N-1).

Star Topology:

In star topology, all the devices are connected to a single hub through a cable. This hub is the central node and all other nodes are connected to the central node. The hub can be passive in nature i.e., not an intelligent hub such as broadcasting devices, at the same time the hub can be intelligent known as an active hub. Active hubs have repeaters in them. In Star Topology, many popular Ethernet LAN protocols are used as CD(Collision Detection), CSMA (Carrier Sense Multiple Access), etc.

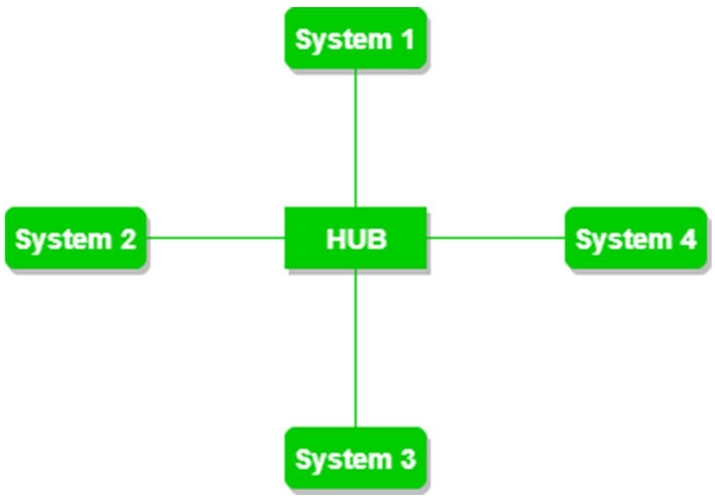


Figure 2: A star topology having four systems connected to a single point of connection i.e. hub.

Ring Topology:

In this topology, it forms a ring connecting devices with exactly two neighboring devices.

A number of repeaters are used for Ring topology with a large number of nodes, because if someone wants to send some data to the last node in the ring topology with 100 nodes, then the data will have to pass through 99 nodes to reach the 100th node. Hence to prevent data loss repeaters are used in the network.

The transmission is unidirectional, but it can be made bidirectional by having 2 connections between each Network Node, it is called Dual Ring Topology. In-Ring Topology, the Token Ring Passing protocol is used by the workstations to transmit the data.

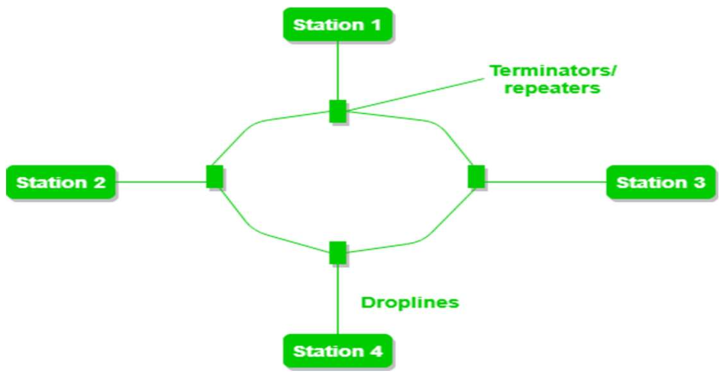
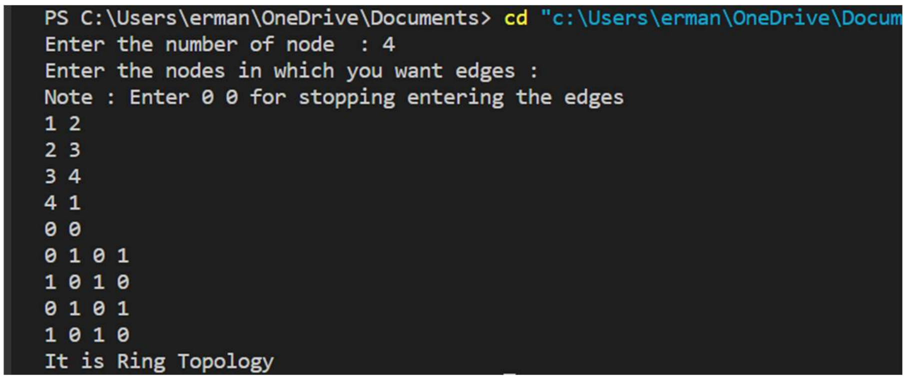


Figure 3: A ring topology comprises 4 stations connected with each forming a ring.

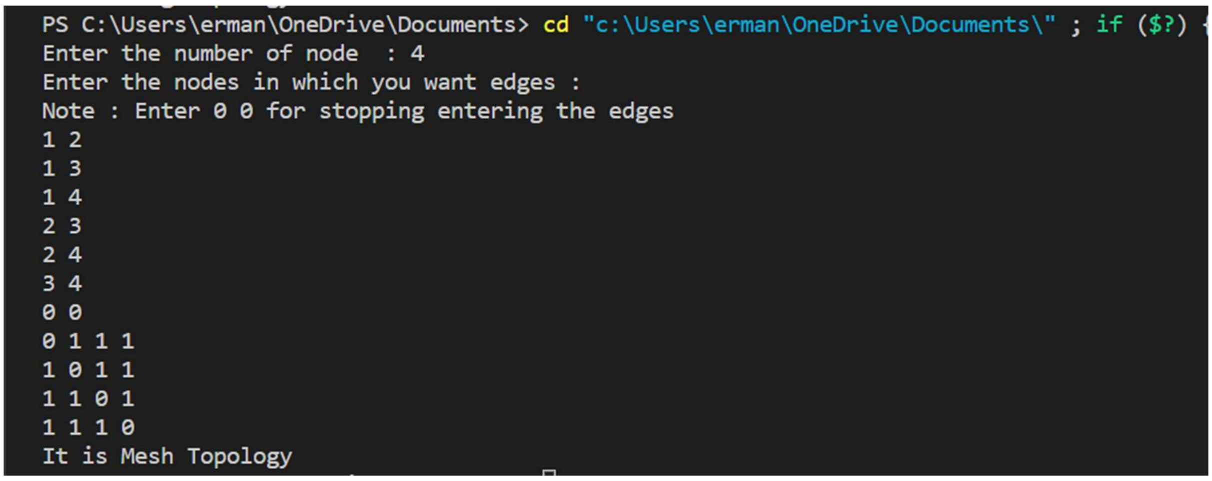
Code:

```
#include<bits/stdc++.h>
using namespace std;

static void topology(vector<vector<int>> &mat,int n){
    bool flag=true;
    int sum = 0,rowSum=0;
    int rowSumMatrix = 0;
    for(int i=0;i<mat[0].size();i++){
        rowSumMatrix += mat[0][i];
     }
    for(int i=0;i<mat.size();i++){
        rowSum = 0;
        for(int j=0;j<mat[i].size();j++){
            sum+=mat[i][j];
            rowSum+=mat[i][j];
        }
        if(rowSum!=rowSumMatrix){
            flag = false;
        }
    }
    int row1Sum = 0,col1Sum = 0;

    for(int i=0;i<n;i++){
        row1Sum += mat[0][i];
        col1Sum += mat[i][0];
    }

    if((sum==2*n-2) and (row1Sum==col1Sum)){
        cout<<"It is Star Topology"<<endl;
    }
    else if(!flag){
        cout<<"Not any topology"<<endl;
    }
    else if(sum==2*n){
        cout<<"It is Ring Topology"<<endl;
    }
    else if(sum==n*(n-1)){
        cout<<"It is Mesh Topology"<<endl;
    }

}

int main(){

    int n;
    cout<<"Enter the number of node  : ";
    cin>>n;
    vector<vector<int>> edges;
    cout<<"Enter the nodes in which you want edges : "<<endl;
    cout<<"Note : Enter 0 0 for stopping entering the edges "<<endl;
    while(1){
        vector<int> a(2);
        int edge;
        for(int j=0;j<2;j++){
            cin>>edge;
            a[j] = edge;
        }
        if(edge==0){
            break;
        }
        edges.push_back(a);
    }
int m = edges.size();

    vector<vector<int>> matrix(n,vector<int>(n,0));

    for(int i=0;i<m;i++){
        for(int j=0;j<m;j++){
            matrix[edges[i][0]-1][edges[i][1]-1] = 1;
            matrix[edges[i][1]-1][edges[i][0]-1] = 1;
        }
    }

    for(int i = 0;i<matrix.size();i++){
        for(int j = 0;j<n;j++){
            cout<<matrix[i][j]<<" ";
        }
        cout<<endl;
    }

    topology(matrix,matrix.size());
    return 0;
}
```

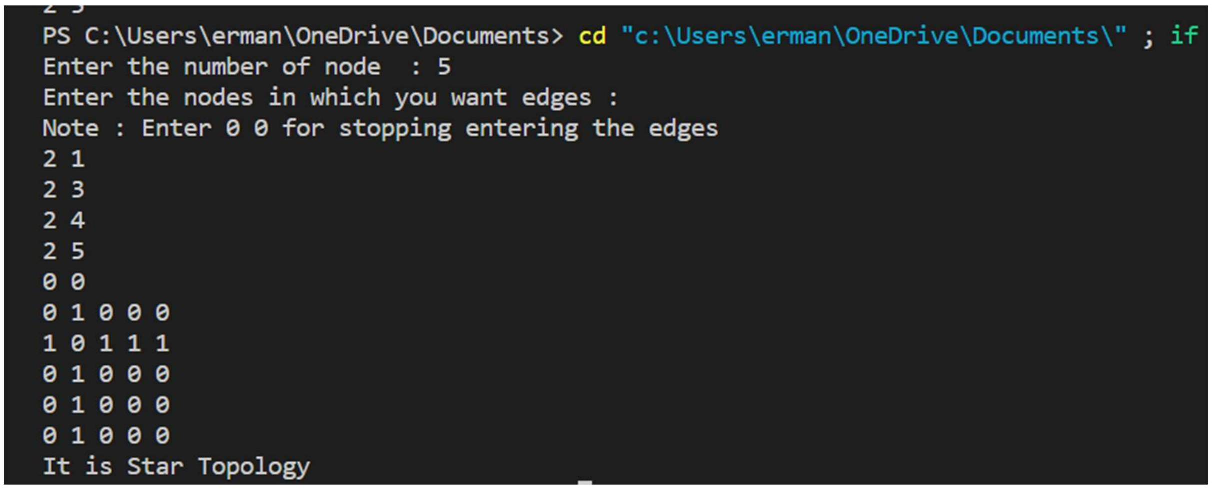
Output:



```
PS C:\Users\erman\OneDrive\Documents> cd "c:\Users\erman\OneDrive\Docum
Enter the number of node  : 4
Enter the nodes in which you want edges :
Note : Enter 0 0 for stopping entering the edges
1 2
2 3
3 4
4 1
0 0
0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 0
It is Ring Topology
```



```
PS C:\Users\erman\OneDrive\Documents> cd "c:\Users\erman\OneDrive\Documents\" ; if ($?) {
Enter the number of node  : 4
Enter the nodes in which you want edges :
Note : Enter 0 0 for stopping entering the edges
1 2
1 3
1 4
2 3
2 4
3 4
0 0
0 1 1 1
1 0 1 1
1 1 0 1
1 1 1 0
It is Mesh Topology
```



```
PS C:\Users\erman\OneDrive\Documents> cd "c:\Users\erman\OneDrive\Documents\" ; if
Enter the number of node  : 5
Enter the nodes in which you want edges :
Note : Enter 0 0 for stopping entering the edges
2 1
2 3
2 4
2 5
0 0
0 1 0 0 0
1 0 1 1 1
0 1 0 0 0
0 1 0 0 0
0 1 0 0 0
It is Star Topology
```

Result: Identified the type of network topology