**1. Write a Program to create a SET A and determine the cardinality of SET for an input array of elements (repetition allowed) and perform the following operations on the SET:**

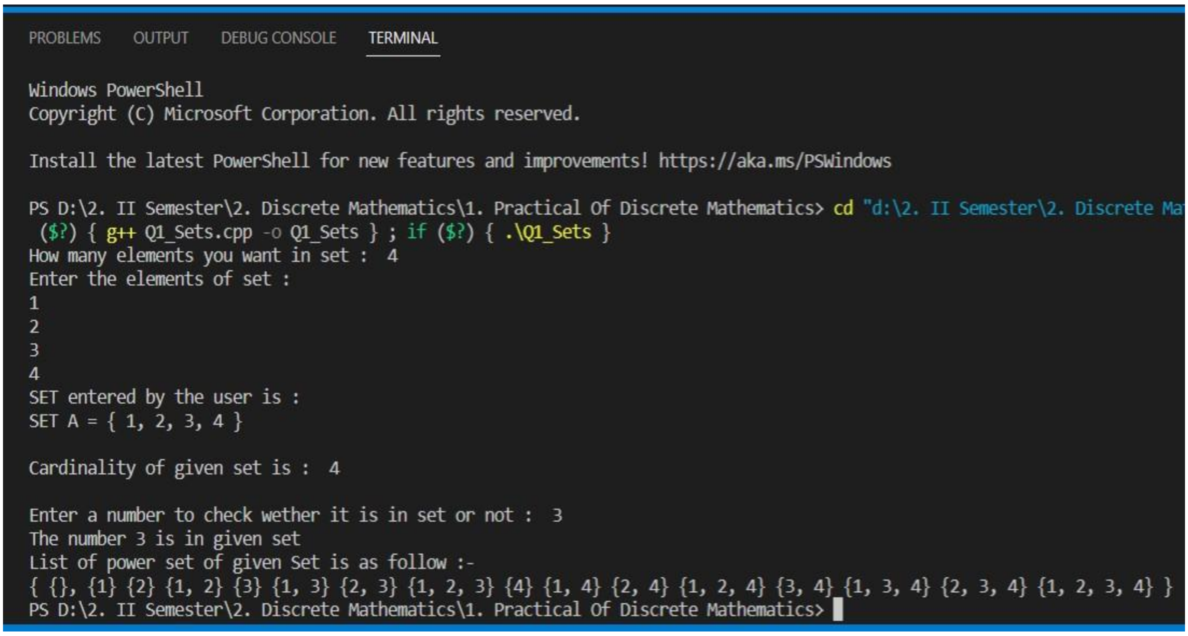
**a) ismember (a, A): check whether an element belongs to set or not and return value as true/false.**

**b) powerset(A): list all the elements of power set of A.**

**->**

```
#include <iostream>

#include <math.h>

using namespace std;


bool checkElement(int arr[], int n)

{

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

    {

        if (arr[i] == n)

            return true;

    }

    return false;

}


void powerset(int arr[], int size)

{

    int count, temp;

    count = pow(2, size);

    cout << "{ {},";

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

    {

        temp = i;

        cout << " {";

        for (int j = 0; j < size; j++)

        {

            if (temp & 1)

                cout << arr[j] << ", ";

            temp = temp >> 1;
        }
        cout << "\b\b}";
    }
    cout << " }";
}

int main()
{
    int n, num;
    cout << "How many elements you want in set :  ";
    cin >> n;
    int arr[n];
    cout << "Enter the elements of set :\n";
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }
    cout << "SET entered by the user is :\nSET A = { ";
    for (int i = 0; i < n; i++)
    {
        cout << arr[i] << ", ";
    }
    cout<<"\b\b }"<<endl;

    cout << "\nCardinality of given set is :  " << n << endl;

    cout << "\nEnter a number to check wether it is in set or not :  ";
    cin >> num;
    if (checkElement(arr, num) == 1)
    {
        cout << "The number " << num << " is in given set" << endl;
    }
    else
        cout << "The number " << num << " is not in the given set" << endl;
    cout << "List of power set of given Set is as follow :-" << endl;

    powerset(arr, n);


    return 0;
}
```

## Output Of 1st Program in IDE :-



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd "d:\2. II Semester\2. Discrete Ma
 ($?) { g++ Q1_Sets.cpp -o Q1_Sets } ; if ($?) { .\Q1_Sets }
How many elements you want in set :  4
Enter the elements of set :
1
2
3
4
SET entered by the user is :
SET A = { 1, 2, 3, 4 }

Cardinality of given set is :  4

Enter a number to check wether it is in set or not :  3
The number 3 is in given set
List of power set of given Set is as follow :-
{ {}, {1} {2} {1, 2} {3} {1, 3} {2, 3} {1, 2, 3} {4} {1, 4} {2, 4} {1, 2, 4} {3, 4} {1, 3, 4} {2, 3, 4} {1, 2, 3, 4} }
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> 
```

**2. Create a class SET and take two sets as input from user to perform following SET Operations:**

**a) Subset: Check whether one set is a subset of other or not.**

**b) Union and Intersection of two Sets.**

**c) Complement: Assume Universal Set as per the input elements from the user.**

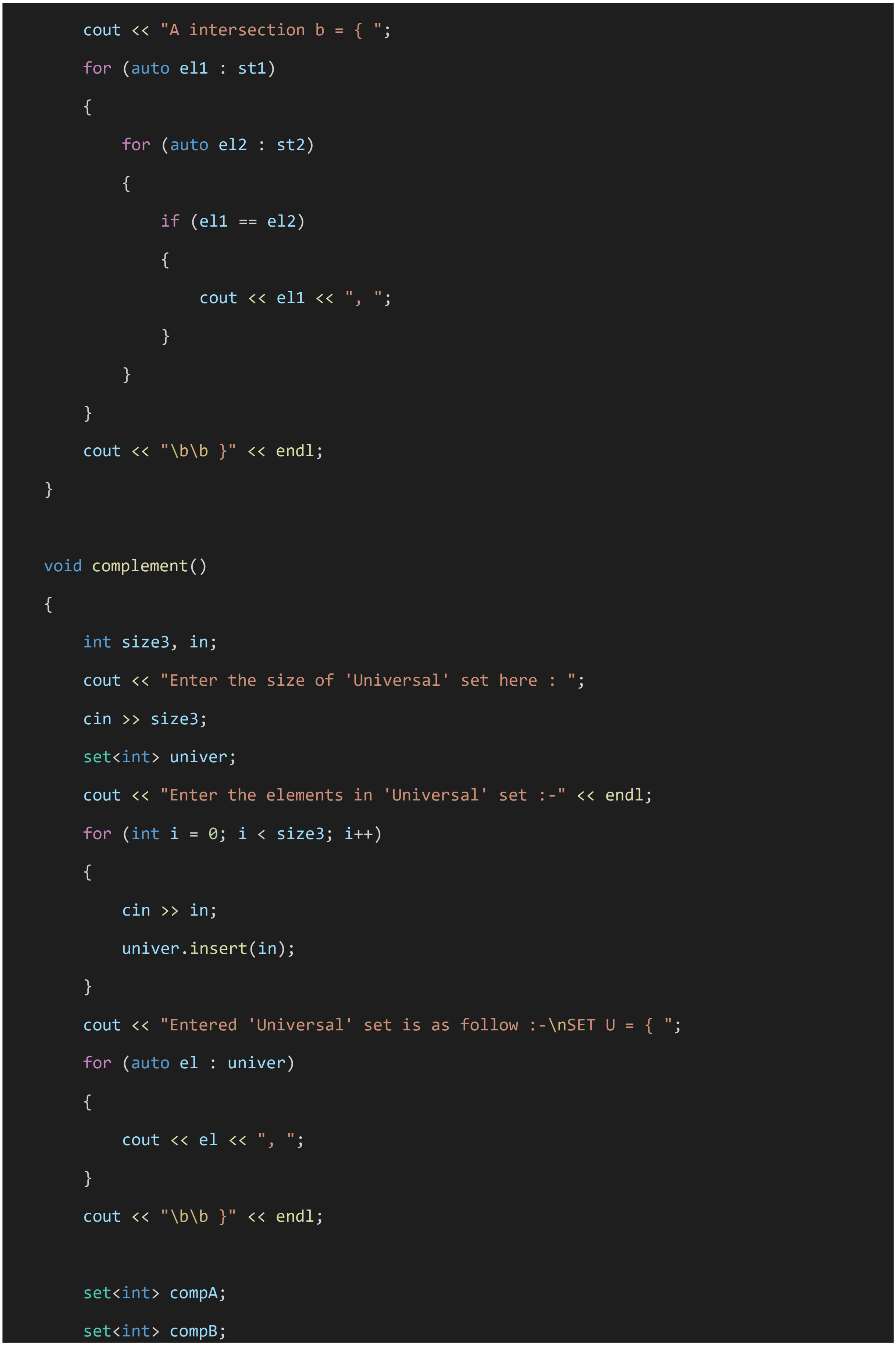
**d) Set Difference and Symmetric Difference between two SETS**

**e) Cartesian Product of Sets.**

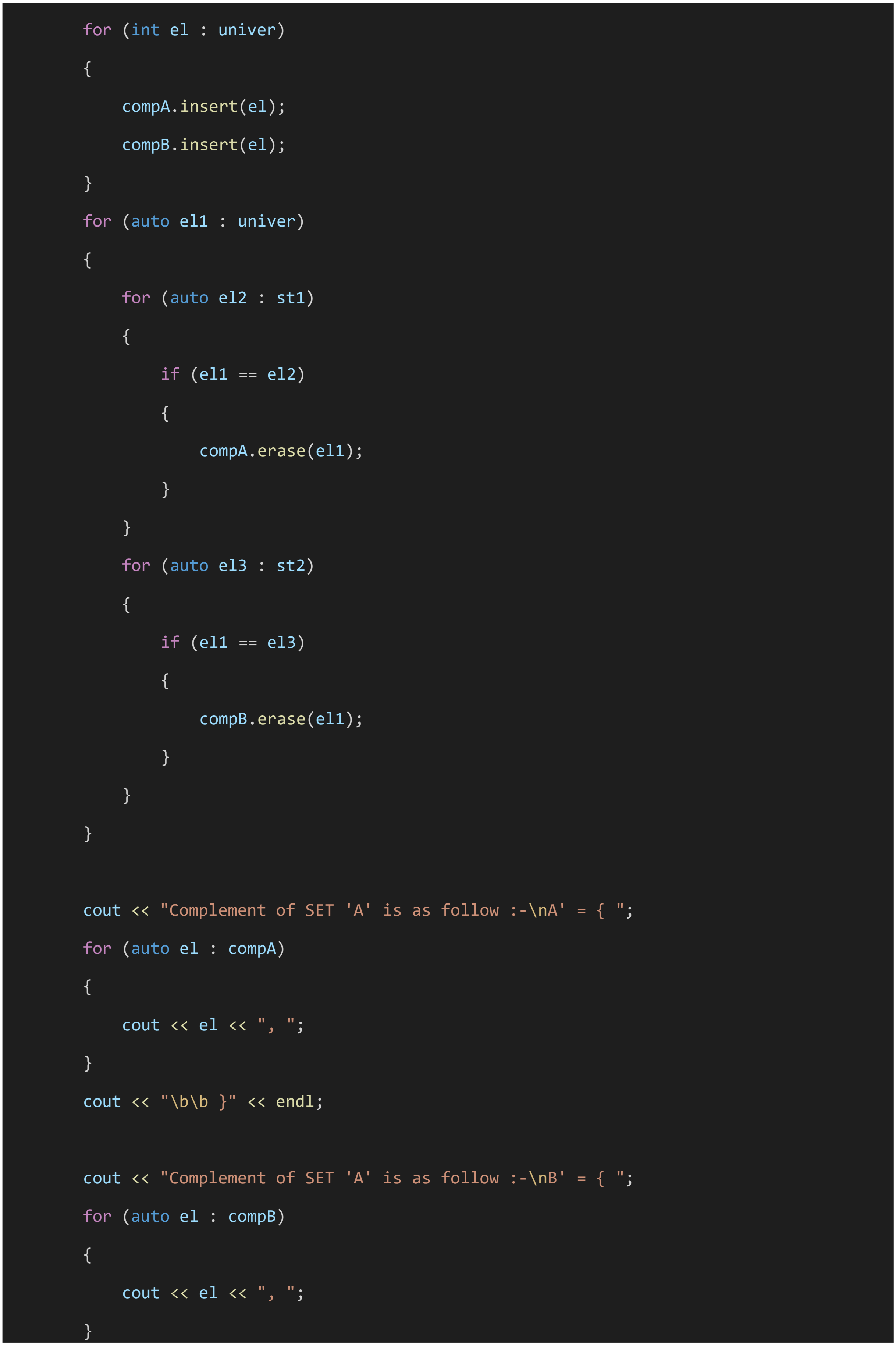
->

```
#include <iostream>

#include <set>

using namespace std;


class SET

{

private:

    set<int> st1;

    set<int> st2;


public:

    void inputSet()

    {

        int size1, size2, e1;

        cout << "Enter the size of SET 'A' here :  ";

        cin >> size1;

        for (int i = 0; i < size1; i++)

        {

            cout << "Enter the element :  ";

            cin >> e1;

            st1.insert(e1);

        }

        cout << "Enter the size of SET 'B' here :  ";

        cin >> size2;

        for (int i = 0; i < size2; i++)

        {

            cout << "Enter the element :  ";

            cin >> e1;

            st2.insert(e1);

        }

        cout << "Set 'A' = { ";

        for (auto el : st1)

        {

            cout << el << ", ";
        }
        cout << "\b\b }" << endl;
        cout << "Set 'B' = { ";
        for (auto el : st2)
        {
            cout << el << ", ";
        }
        cout << "\b\b }" << endl;
    }

    void subSet()
    {
        int ctr1 = 0, ctr2 = 0;
        for (int emt1 : st1)
        {
            for (int emt2 : st2)
            {
                if (emt1 == emt2)
                {
                    ctr1++;
                    break;
                }
            }
        }
        for (int emt2 : st1)
        {
            for (int emt1 : st2)
            {
                if (emt2 == emt1)
                {
                    ctr2++;
                    break;
                }
            }
        }
```

```
        if (ctr1 == st1.size())
        {
            cout << "SET 'A' is subset of SET 'B'" << endl;
        }
        if (ctr2 == st2.size())
        {
            cout << "SET 'B' is subset of SET 'A'" << endl;
        }
        if (ctr1 != st1.size() && ctr2 != st2.size())
        {
            cout << "Neither SET 'A' is subset of SET 'B'\nnor SET 'B' is subset of SET 'A'" << endl;
        }
    }

    void setUniItc()
    {
        set<int> uni;
        for (auto el : st1)
        {
            uni.insert(el);
        }
        for (auto el : st2)
        {
            uni.insert(el);
        }
        cout << "Union of set A & B is as follow :-" << endl;
        cout << "A U B = { ";
        for (auto el : uni)
        {
            cout << el << ", ";
        }
        cout << "\b\b }" << endl;

        cout << "Intersection of set A & B is as follow :-" << endl;
```



```
    cout << "A intersection b = { ";
    for (auto el1 : st1)
    {
        for (auto el2 : st2)
        {
            if (el1 == el2)
            {
                cout << el1 << ", ";
            }
        }
    }
    cout << "\b\b }" << endl;
}

void complement()
{
    int size3, in;
    cout << "Enter the size of 'Universal' set here : ";
    cin >> size3;
    set<int> univer;
    cout << "Enter the elements in 'Universal' set :-" << endl;
    for (int i = 0; i < size3; i++)
    {
        cin >> in;
        univer.insert(in);
    }
    cout << "Entered 'Universal' set is as follow :-\nSET U = { ";
    for (auto el : univer)
    {
        cout << el << ", ";
    }
    cout << "\b\b }" << endl;

    set<int> compA;
    set<int> compB;
```

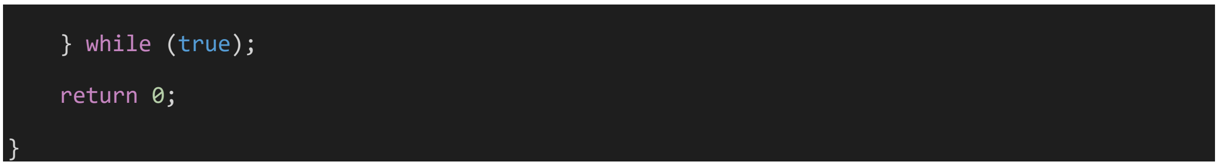


```
        for (int el : univer)
        {
            compA.insert(el);
            compB.insert(el);
        }
        for (auto el1 : univer)
        {
            for (auto el2 : st1)
            {
                if (el1 == el2)
                {
                    compA.erase(el1);
                }
            }
            for (auto el3 : st2)
            {
                if (el1 == el3)
                {
                    compB.erase(el1);
                }
            }
        }

        cout << "Complement of SET 'A' is as follow :-\nA' = { ";
        for (auto el : compA)
        {
            cout << el << ", ";
        }
        cout << "\b\b }" << endl;

        cout << "Complement of SET 'A' is as follow :-\nB' = { ";
        for (auto el : compB)
        {
            cout << el << ", ";
        }
```

```
    cout << "\b\b }" << endl;
}

void setsymdiffer()
{
    set<int> ab;
    set<int> ba;
    for (auto el : st1)
    {
        ab.insert(el);
    }
    for (auto el : st2)
    {
        ba.insert(el);
    }
    for (auto el1 : st1)
    {
        for (auto el2 : st2)
        {
            if (el1 == el2)
            {
                ab.erase(el1);
                ba.erase(el1);
            }
        }
    }
    cout << "Set Difference :-\nA - B = { ";
    for (auto el : ab)
    {
        cout << el << ", ";
    }
    cout << "\b\b }" << endl;

    cout << "Set Difference :-\nB - A = { ";
    for (auto el : ba)
```

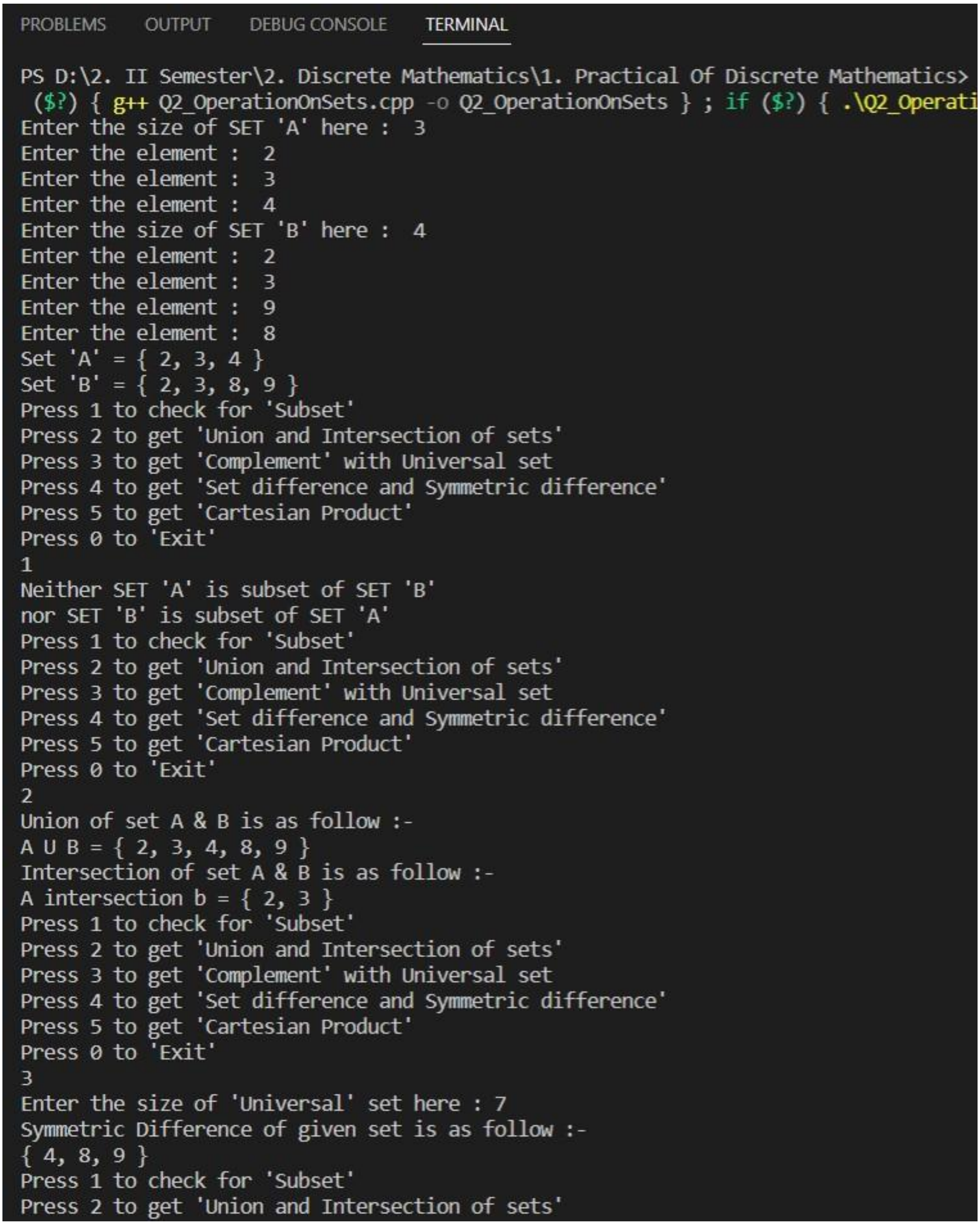
```
        {
            cout << el << ", ";
        }
        cout << "\b\b }" << endl;
        set<int> symD;
        for (auto el : ab)
        {
            symD.insert(el);
        }
        for (auto el : ba)
        {
            symD.insert(el);
        }
        cout << "Symmetric Difference of given set is as follow :-\n{ ";
        for (auto el : symD)
        {
            cout << el << ", ";
        }
        cout << "\b\b }" << endl;
    }

    void carProd()
    {
        cout << "\nCartesian Product of given Set 'A' and 'B' is as follow :-" << endl;
        for (auto el1 : st1)
        {
            for (auto el2 : st2)
            {
                cout << "(" << el1 << ", " << el2 << ")"
                     << " ";
            }
        }
        cout << endl;
    }
};
```

```
int main()
{
    SET s1;
    s1.inputSet();
    do
    {
        int ch;
        cout << "Press 1 to check for 'Subset'" << endl;
        cout << "Press 2 to get 'Union and Intersection of sets'" << endl;
        cout << "Press 3 to get 'Complement' with Universal set" << endl;
        cout << "Press 4 to get 'Set difference and Symmetric difference'" << endl;
        cout << "Press 5 to get 'Cartesian Product'" << endl;
        cout << "Press 0 to 'Exit'" << endl;
        cin >> ch;
        switch (ch)
        {
        case 1:
            s1.subSet();
            break;
        case 2:
            s1.setUniItc();
            break;
        case 3:
            s1.complement();
            break;
        case 4:
            s1.setsymdiffer();
            break;
        case 5:
            s1.carProd();
            break;
        default:
            exit(0);
        }
```

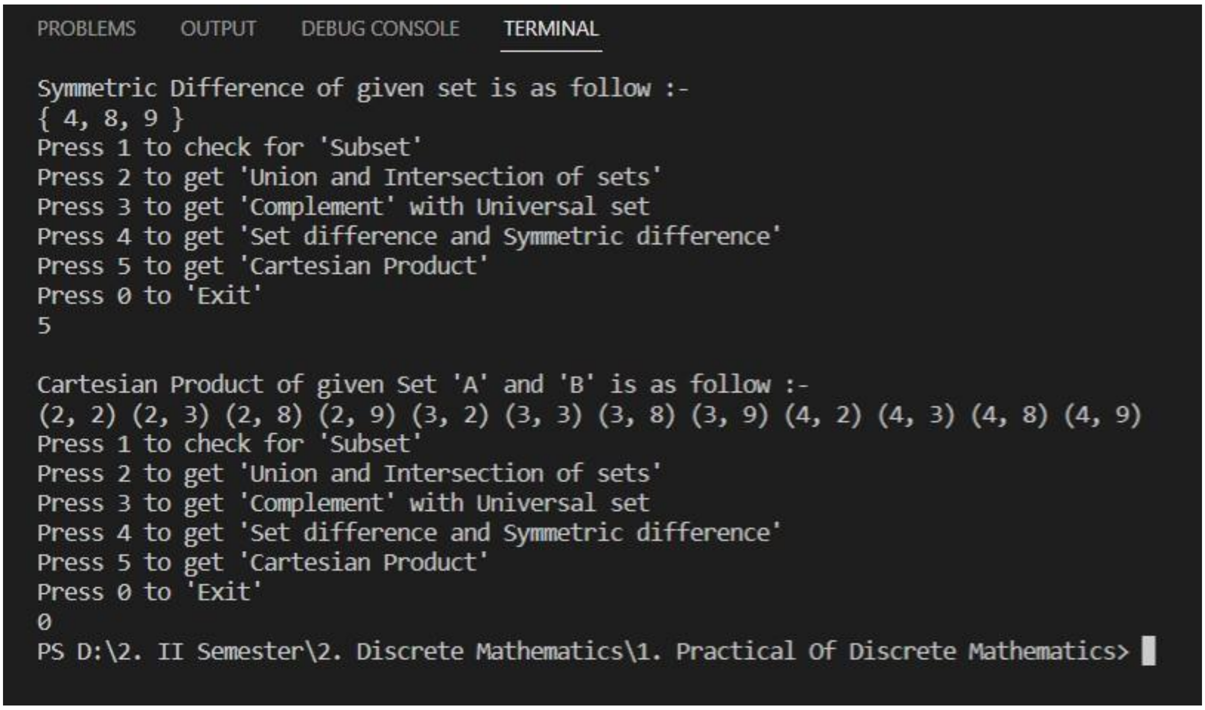


```
    } while (true);

    return 0;
}
```

## Output Of 2$^{nd}$ Program in IDE :-



```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics>
 ($?) { g++ Q2_OperationOnSets.cpp -o Q2_OperationOnSets } ; if ($?) { .\Q2_Operati
Enter the size of SET 'A' here :  3
Enter the element :  2
Enter the element :  3
Enter the element :  4
Enter the size of SET 'B' here :  4
Enter the element :  2
Enter the element :  3
Enter the element :  9
Enter the element :  8
Set 'A' = { 2, 3, 4 }
Set 'B' = { 2, 3, 8, 9 }
Press 1 to check for 'Subset'
Press 2 to get 'Union and Intersection of sets'
Press 3 to get 'Complement' with Universal set
Press 4 to get 'Set difference and Symmetric difference'
Press 5 to get 'Cartesian Product'
Press 0 to 'Exit'
1
Neither SET 'A' is subset of SET 'B'
nor SET 'B' is subset of SET 'A'
Press 1 to check for 'Subset'
Press 2 to get 'Union and Intersection of sets'
Press 3 to get 'Complement' with Universal set
Press 4 to get 'Set difference and Symmetric difference'
Press 5 to get 'Cartesian Product'
Press 0 to 'Exit'
2
Union of set A & B is as follow :-
A U B = { 2, 3, 4, 8, 9 }
Intersection of set A & B is as follow :-
A intersection b = { 2, 3 }
Press 1 to check for 'Subset'
Press 2 to get 'Union and Intersection of sets'
Press 3 to get 'Complement' with Universal set
Press 4 to get 'Set difference and Symmetric difference'
Press 5 to get 'Cartesian Product'
Press 0 to 'Exit'
3
Enter the size of 'Universal' set here : 7
Symmetric Difference of given set is as follow :-
{ 4, 8, 9 }
Press 1 to check for 'Subset'
Press 2 to get 'Union and Intersection of sets'
```



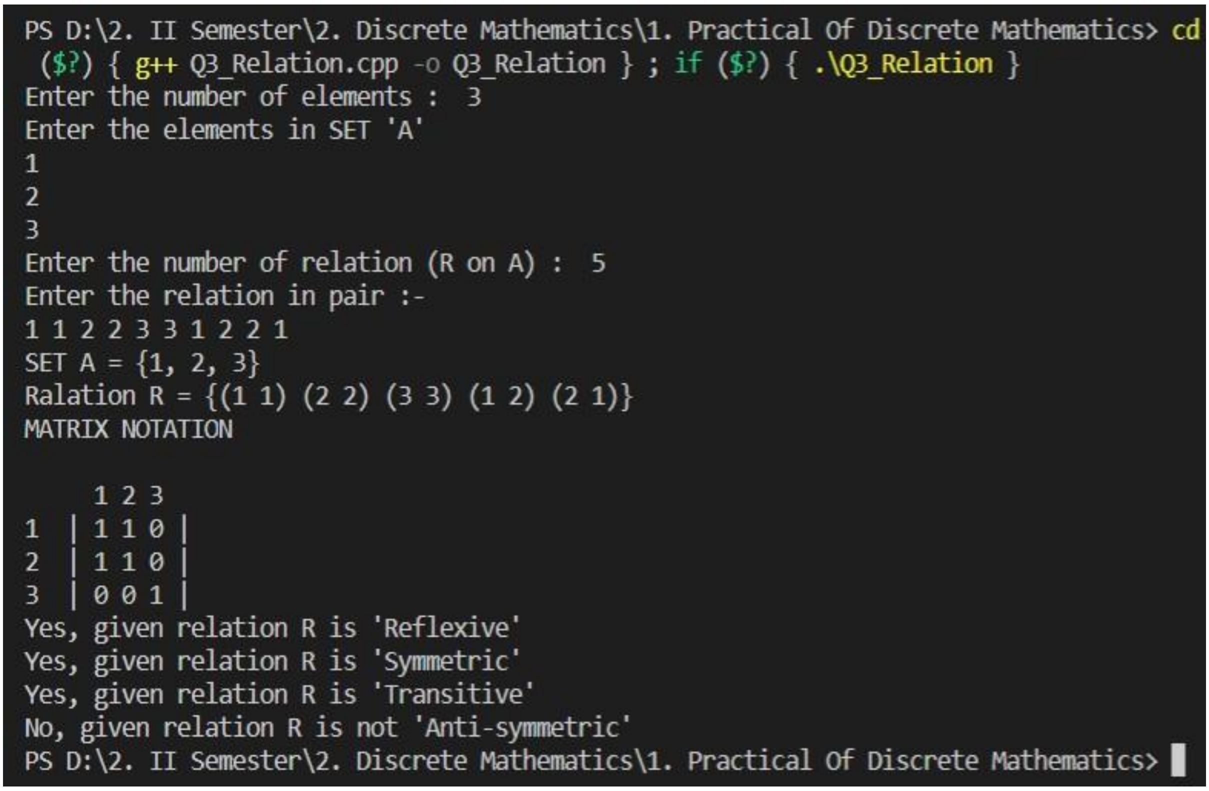
```
Symmetric Difference of given set is as follow :-
{ 4, 8, 9 }
Press 1 to check for 'Subset'
Press 2 to get 'Union and Intersection of sets'
Press 3 to get 'Complement' with Universal set
Press 4 to get 'Set difference and Symmetric difference'
Press 5 to get 'Cartesian Product'
Press 0 to 'Exit'
5

Cartesian Product of given Set 'A' and 'B' is as follow :-
(2, 2) (2, 3) (2, 8) (2, 9) (3, 2) (3, 3) (3, 8) (3, 9) (4, 2) (4, 3) (4, 8) (4, 9)
Press 1 to check for 'Subset'
Press 2 to get 'Union and Intersection of sets'
Press 3 to get 'Complement' with Universal set
Press 4 to get 'Set difference and Symmetric difference'
Press 5 to get 'Cartesian Product'
Press 0 to 'Exit'
0
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics>
```

**3. Create a class RELATION, use Matrix notation to represent a relation. Include functions to check if the relation is Reflexive, Symmetric, Anti-symmetric and Transitive. Write a Program to use this class.**

**->**

```
#include <iostream>

using namespace std;


class RELATION

{

private:

    int stSize, rlSize, *setA, *relR, **mtx;


public:

    void emptySet();

    int inpSet();

    void inpRel();

    void printSet();

    void printRel();
    void matrix();
    void reflexive();
    void symmetric();
    void transitive();
    void antiSymmetric();
};

void RELATION ::emptySet()
{
    cout << "SET is empty" << endl;
    cout << "SET A = { }" << endl;
    cout << "SET A has no element.\nHence, relation R is also empty." << endl;
    cout << "Relation R = { }\nTherefore, no 'Matrix Notation'" << endl;
    cout << "Relation R is not 'Reflexive'" << endl;
    symmetric();
    transitive();
    antiSymmetric();
}

int RELATION ::inpSet()
{
    cout << "Enter the number of elements :  ";
    cin >> stSize;
    if (stSize == 0)
        return 0;
    setA = new int[stSize];
    cout << "Enter the elements in SET 'A'" << endl;
    for (int i = 0; i < stSize; i++)
    {
        cin >> setA[i];
    }
}

void RELATION ::inpRel()
{
    cout << "Enter the number of relation (R on A) :  ";
    cin >> rlSize;
    relR = new int[2 * rlSize];
    cout << "Enter the relation in pair :-" << endl;
    for (int i = 0; i < (2 * rlSize); i++)
    {
        cin >> relR[i];
    }
}

void RELATION ::printSet()
{
    cout << "SET A = {";
    for (int i = 0; i < stSize; i++)
    {
        cout << setA[i] << ", ";
    }
    cout << "\b\b}" << endl;
}

void RELATION ::printRel()
{
    cout << "Ralation R = {";
    for (int i = 0; i < (2 * rlSize); i++)
    {
        if (i % 2 == 0)
        {
            cout << "(";
        }
        cout << relR[i] << " ";
        if (i % 2 != 0)
        {
            cout << "\b) ";
        }
    }
    cout << "\b}" << endl;
}

void RELATION ::matrix()
{
    int idx1, idx2;
    mtx = new int *[stSize];
    for (int i = 0; i < stSize; i++)
    {
        mtx[i] = new int[stSize];
    }
    for (int i = 0; i < stSize; i++)
    {
        for (int j = 0; j < stSize; j++)
        {
            mtx[i][j] = 0;
        }
    }
    for (int i = 0; i < (2 * rlSize); i += 2)
    {
        for (int j = 0; j < stSize; j++)
        {
            if (relR[i] == setA[j])
            {
                idx1 = j;
                break;
            }
        }
        for (int k = 0; k < stSize; k++)
        {
            if (relR[i + 1] == setA[k])
            {
                idx2 = k;
                break;
            }
        }
        mtx[idx1][idx2] = 1;
    }

    cout << "MATRIX NOTATION\n\n";
    cout << "     ";
    for (int i = 0; i < stSize; i++)
    {
        cout << setA[i] << " ";
    }
    cout << endl;
    for (int i = 0; i < stSize; i++)
    {
        cout << setA[i] << "  | ";
        for (int j = 0; j < stSize; j++)
        {
            cout << mtx[i][j] << " ";
        }
        cout << "|" << endl;
    }
}

void RELATION ::reflexive()
{
    int ctr = 0;
    for (int i = 0; i < stSize; i++)
    {
        if (mtx[i][i] == 1)
        {
            ctr++;
        }
    }
    if (ctr == stSize)
    {
        cout << "Yes, given relation R is 'Reflexive'" << endl;
    }
    else
        cout << "No, given relation R is not 'Reflexive'" << endl;
}

void RELATION ::symmetric()
{
    int ctr = 0;
    for (int i = 0; i < stSize; i++)
    {
        for (int j = 0; j < stSize; j++)
        {
            if (mtx[i][j] == mtx[j][i])
            {
                continue;
            }
            else
            {
                ctr++;
                break;
            }
        }
    }
    if (ctr != 0)
    {
        cout << "No, given relation R is not 'Symmetric'" << endl;
    }
    else
        cout << "Yes, given relation R is 'Symmetric'" << endl;
}

void RELATION ::transitive()
{
    int ctr = 0;
    for (int i = 0; i < stSize; i++)
    {
        for (int j = 0; j < stSize; j++)
        {
            if (mtx[i][j] == 1)
            {
                for (int k = 0; k < stSize; k++)
                {
                    if (mtx[j][k] == 1)
                    {
                        if (mtx[i][k] == 1)
                        {
                            continue;
                        }
                        else
                        {
                            ctr++;
                            break;
                        }
                    }
                }
            }
        }
    }
    if (ctr != 0)
    {
        cout << "No, given relation R is not 'Transitive'" << endl;
    }
    else
        cout << "Yes, given relation R is 'Transitive'" << endl;
}

void RELATION ::antiSymmetric()
{
    int flag = 0;
    for (int i = 0; i < stSize; i++)
    {
        for (int j = 0; j < stSize; j++)
        {
            if (i != j)
            {
                if (mtx[i][j] == 1 && mtx[i][j] != mtx[j][i])
                {
                    continue;
                }
                else if (mtx[i][j] == 1)
                {
                    flag = 1;
                    break;
                }
            }
        }
    }
    if (flag == 0)
    {
        cout << "Yes, given relation R is 'Anti-symmetric'" << endl;
    }
    else
        cout << "No, given relation R is not 'Anti-symmetric'" << endl;
}

int main()
{
    RELATION re;
    int r = re.inpSet();
    if (r == 0)
    {
        re.emptySet();
    }
    else
    {
        re.inpRel();

        re.printSet();

        re.printRel();

        re.matrix();

        re.reflexive();

        re.symmetric();

        re.transitive();

        re.antiSymmetric();

    }


    return 0;

}
```

## Output Of 3rd Program in IDE :-



```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd
 ($?) { g++ Q3_Relation.cpp -o Q3_Relation } ; if ($?) { .\Q3_Relation }
Enter the number of elements :  3
Enter the elements in SET 'A'
1
2
3
Enter the number of relation (R on A) :  5
Enter the relation in pair :-
1 1 2 2 3 3 1 2 2 1
SET A = {1, 2, 3}
Ralation R = {(1 1) (2 2) (3 3) (1 2) (2 1)}
MATRIX NOTATION

     1 2 3
1  | 1 1 0 |
2  | 1 1 0 |
3  | 0 0 1 |
Yes, given relation R is 'Reflexive'
Yes, given relation R is 'Symmetric'
Yes, given relation R is 'Transitive'
No, given relation R is not 'Anti-symmetric'
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> 
```

**4. Use the functions defined in Ques 3 to check whether the given relation is:**

**a) Equivalent, or**

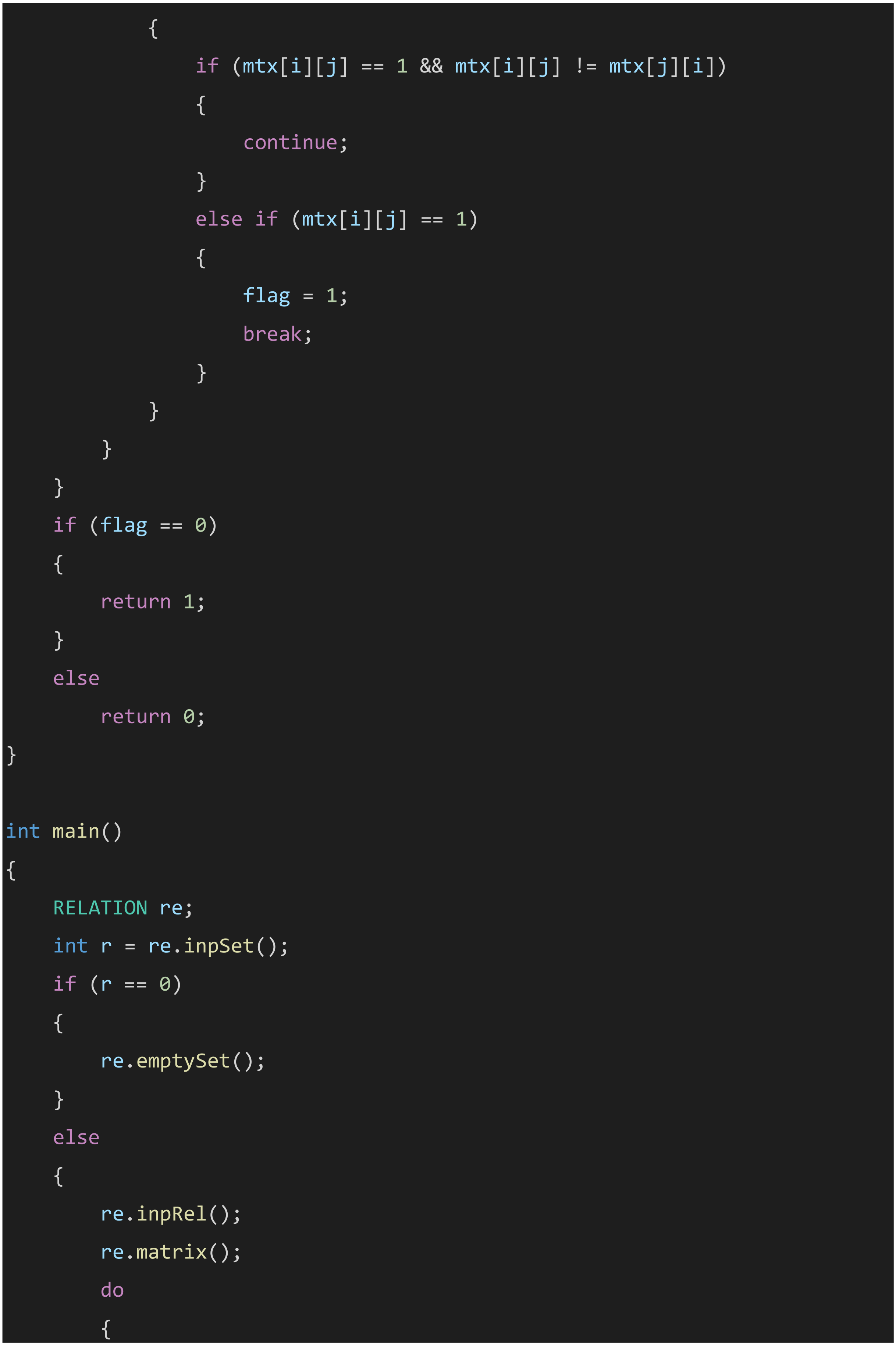
**b) Partial Order relation, or**

**c) None**

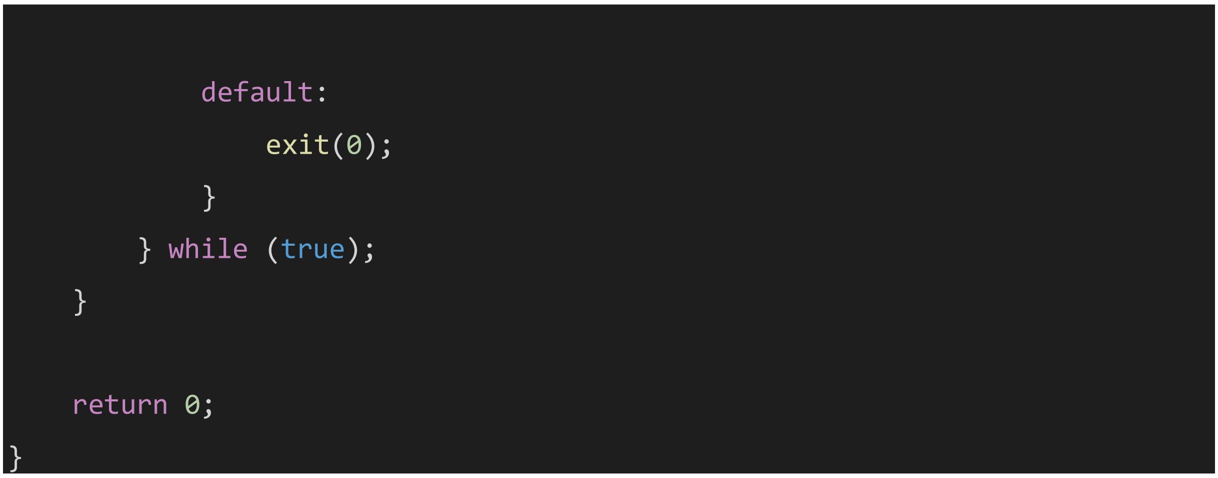
**->**

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

class RELATION
{
private:
    int stSize, rlSize, *setA, *relR, **mtx;

public:
    void emptySet();
    int inpSet();
    void inpRel();
    void matrix();
    int reflexive();
    int symmetric();
    int transitive();
    int antiSymmetric();
};

void RELATION ::emptySet()
{
    cout << "SET is empty" << endl;
    cout << "SET A = { }" << endl;
    cout << "SET A has no element.\nHence, relation R is also empty." <<
endl;
    cout << "Relation R = { }\nTherefore, no 'Matrix Notation'" << endl;
    cout << "Relation R is not 'Reflexive'" << endl;
    symmetric();
    transitive();
    antiSymmetric();
}

int RELATION ::inpSet()
{
    cout << "Enter the number of elements : ";
    cin >> stSize;
    if (stSize == 0)
        return 0;
    setA = new int[stSize];
    cout << "Enter the elements in SET 'A'" << endl;
    for (int i = 0; i < stSize; i++)
    {
        cin >> setA[i];
    }

    cout << "SET A = {";
    for (int i = 0; i < stSize; i++)
    {
        cout << setA[i] << ", ";
    }
    cout << "\b\b}" << endl;
}

void RELATION ::inpRel()
{
    cout << "Enter the number of relation (R on A) : ";
    cin >> rlSize;
    relR = new int[2 * rlSize];
    cout << "Enter the relation in pair :-" << endl;
    for (int i = 0; i < (2 * rlSize); i++)
    {
        cin >> relR[i];
    }
    cout << "Ralation R = {";
    for (int i = 0; i < (2 * rlSize); i++)
    {
        if (i % 2 == 0)
        {
            cout << "(";
        }
        cout << relR[i] << " ";
        if (i % 2 != 0)
        {
            cout << "\b) ";
        }
    }
    cout << "\b}" << endl;
}

void RELATION ::matrix()
{
    int idx1, idx2;
    mtx = new int *[stSize];
    for (int i = 0; i < stSize; i++)
    {
        mtx[i] = new int[stSize];
    }
    for (int i = 0; i < stSize; i++)
    {
        for (int j = 0; j < stSize; j++)
        {
            mtx[i][j] = 0;
        }
    }
    for (int i = 0; i < (2 * rlSize); i += 2)
    {
        for (int j = 0; j < stSize; j++)
        {
            if (relR[i] == setA[j])
            {
                idx1 = j;
                break;
            }
        }
        for (int k = 0; k < stSize; k++)
        {
            if (relR[i + 1] == setA[k])
            {
                idx2 = k;
                break;
            }
        }
        mtx[idx1][idx2] = 1;
    }
}

int RELATION ::reflexive()
{
    int ctr = 0;
    for (int i = 0; i < stSize; i++)
    {
        if (mtx[i][i] == 1)
        {
            ctr++;
        }
    }
    if (ctr == stSize)
    {
        return 1;
    }
    else
        return 0;
}
```

```
int RELATION ::symmetric()
{
    int ctr = 0;
    for (int i = 0; i < stSize; i++)
    {
        for (int j = 0; j < stSize; j++)
        {
            if (mtx[i][j] == mtx[j][i])
            {
                continue;
            }
            else
            {
                ctr++;
                break;
            }
        }
    }
    if (ctr != 0)
    {
        return 0;
    }
    else
        return 1;
}

int RELATION ::transitive()
{
    int ctr = 0;
    for (int i = 0; i < stSize; i++)
    {
        for (int j = 0; j < stSize; j++)
        {
            if (mtx[i][j] == 1)
            {
                for (int k = 0; k < stSize; k++)
                {
                    if (mtx[j][k] == 1)
                    {
                        if (mtx[i][k] == 1)
                        {
                            continue;
                        }
                        else
                        {
                            ctr++;
                            break;
                        }
                    }
                }
            }
        }
    }
    if (ctr != 0)
    {
        return 0;
    }
    else
        return 1;
}

int RELATION ::antiSymmetric()
{
    int flag = 0;
    for (int i = 0; i < stSize; i++)
    {
        for (int j = 0; j < stSize; j++)
        {
            if (i != j)
```

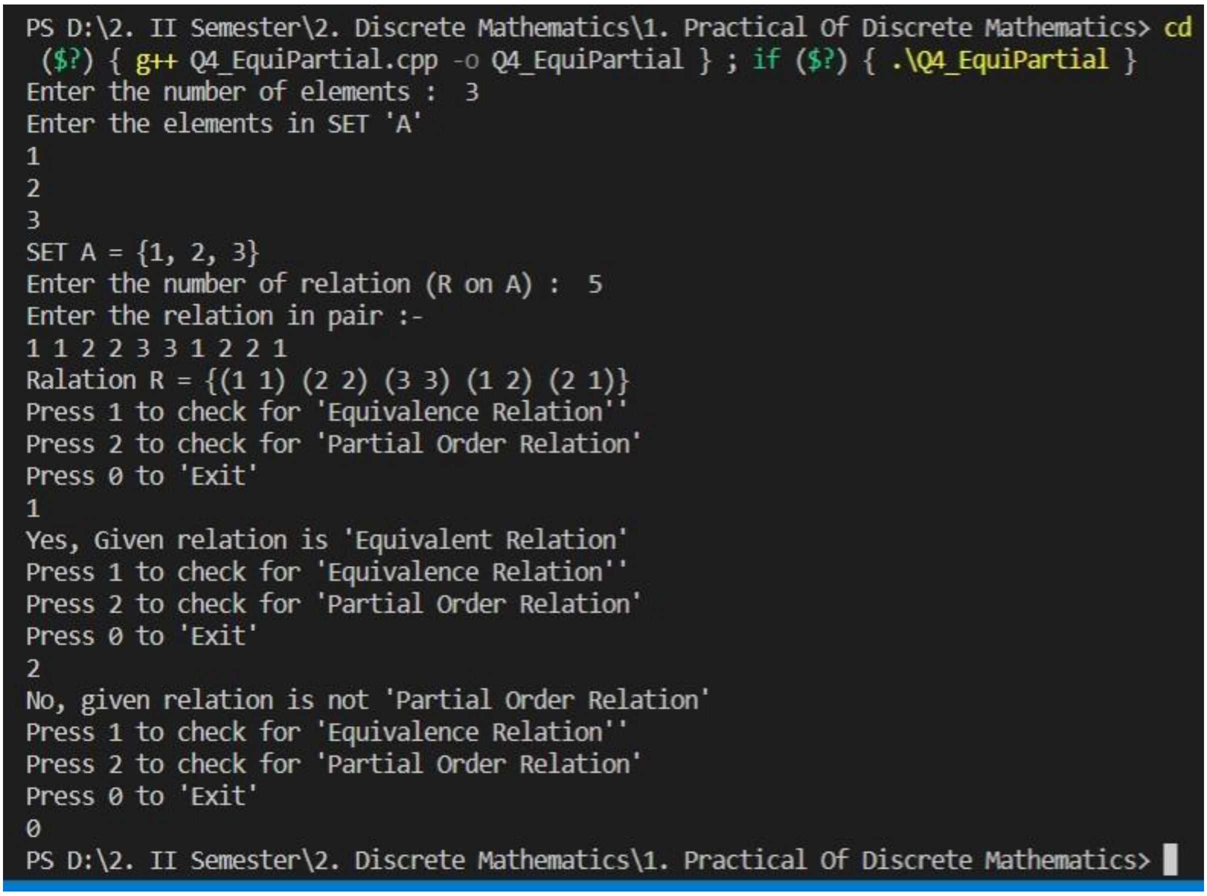


```
            {
                if (mtx[i][j] == 1 && mtx[i][j] != mtx[j][i])
                {
                    continue;
                }
                else if (mtx[i][j] == 1)
                {
                    flag = 1;
                    break;
                }
            }
        }
    }
    if (flag == 0)
    {
        return 1;
    }
    else
        return 0;
}

int main()
{
    RELATION re;
    int r = re.inpSet();
    if (r == 0)
    {
        re.emptySet();
    }
    else
    {
        re.inpRel();
        re.matrix();
        do
        {
            int ch;
            cout << "Press 1 to check for 'Equivalence Relation''" <<
endl;
            cout << "Press 2 to check for 'Partial Order Relation'" <<
endl;
            cout << "Press 0 to 'Exit'" << endl;
            cin >> ch;
            switch (ch)
            {
            case 1:
            {
                if (re.reflexive() && re.symmetric() && re.transitive())
                {
                    cout << "Yes, Given relation is 'Equivalent Relation'"
<< endl;
                }
                else
                    cout << "No, given relation is not 'Equivalence
Relation'" << endl;
            }
            break;

            case 2:
            {
                if (re.reflexive() && re.antiSymmetric() &&
re.transitive())
                {
                    cout << "Yes, Given relation is 'Partial Order
Relation'" << endl;
                }
                else
                    cout << "No, given relation is not 'Partial Order
Relation'" << endl;
            }
            break;
```



```
            default:
                exit(0);
            }
        } while (true);
    }

    return 0;
}
```

## Output Of 4th Program in IDE :-

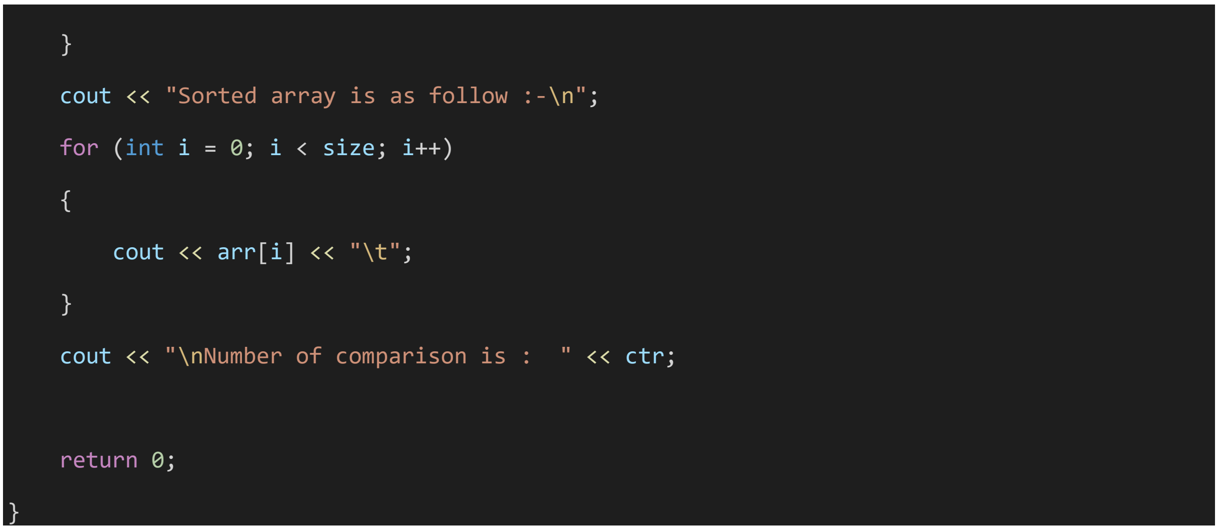


```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd
 ($?) { g++ Q4_EquiPartial.cpp -o Q4_EquiPartial } ; if ($?) { .\Q4_EquiPartial }
Enter the number of elements :  3
Enter the elements in SET 'A'
1
2
3
SET A = {1, 2, 3}
Enter the number of relation (R on A) :  5
Enter the relation in pair :-
1 1 2 2 3 3 1 2 2 1
Ralation R = {(1 1) (2 2) (3 3) (1 2) (2 1)}
Press 1 to check for 'Equivalence Relation''
Press 2 to check for 'Partial Order Relation'
Press 0 to 'Exit'
1
Yes, Given relation is 'Equivalent Relation'
Press 1 to check for 'Equivalence Relation''
Press 2 to check for 'Partial Order Relation'
Press 0 to 'Exit'
2
No, given relation is not 'Partial Order Relation'
Press 1 to check for 'Equivalence Relation''
Press 2 to check for 'Partial Order Relation'
Press 0 to 'Exit'
0
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> 
```

**5. Write a Program to implement Bubble Sort. Find the number of comparisons during each pass and display the intermediate result.**

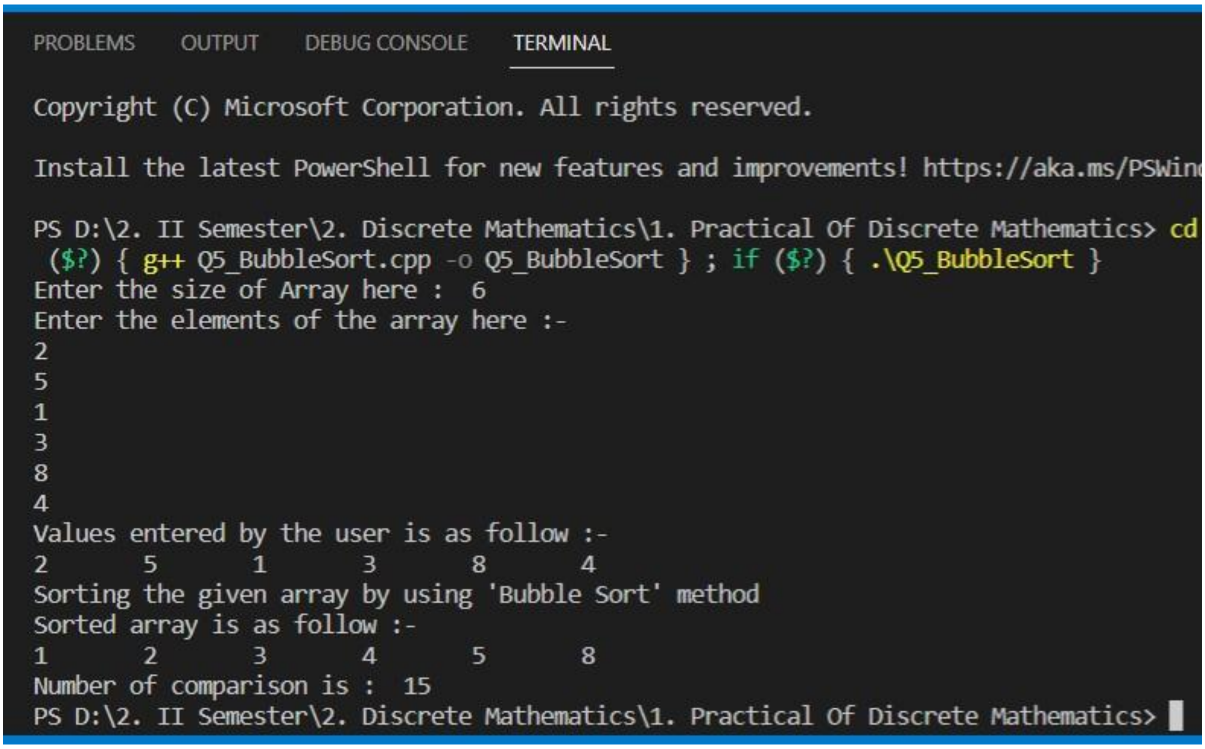
**->**

```
#include <iostream>

using namespace std;


int main()

{

    int size, ctr = 0;

    cout << "Enter the size of Array here :  ";

    cin >> size;

    int arr[size];

    cout << "Enter the elements of the array here :-\n";

    for (int i = 0; i < size; i++)

    {

        cin >> arr[i];

    }

    cout << "Values entered by the user is as follow :-\n";

    for (int i = 0; i < size; i++)

    {

        cout << arr[i] << "\t";

    }

    cout << "\nSorting the given array by using 'Bubble Sort' method\n";

    for (int i = 0; i < size; i++)

    {

        for (int j = 0; j < size - (i + 1); j++)

        {

            if (arr[j] > arr[j + 1])

            {

                arr[j] = arr[j] + arr[j + 1];

                arr[j + 1] = arr[j] - arr[j + 1];

                arr[j] = arr[j] - arr[j + 1];

            }

            ctr++;

        }
```



```
    }
    cout << "Sorted array is as follow :-\n";
    for (int i = 0; i < size; i++)
    {
        cout << arr[i] << "\t";
    }
    cout << "\nNumber of comparison is :  " << ctr;

    return 0;
}
```

## Output Of 5th Program in IDE :-

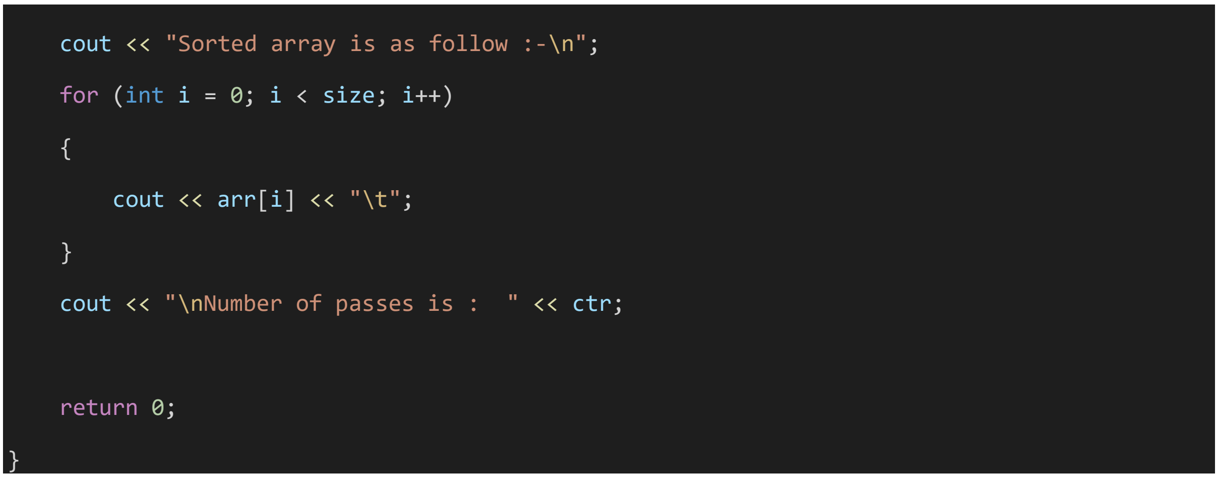


```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWin

PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd
 ($?) { g++ Q5_BubbleSort.cpp -o Q5_BubbleSort } ; if ($?) { .\Q5_BubbleSort }
Enter the size of Array here :  6
Enter the elements of the array here :-
2
5
1
3
8
4
Values entered by the user is as follow :-
2       5       1       3       8       4
Sorting the given array by using 'Bubble Sort' method
Sorted array is as follow :-
1       2       3       4       5       8
Number of comparison is :  15
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> █
```

**6. Write a Program to implement Insertion Sort. Find the number of comparisons during each pass and display the intermediate result.**

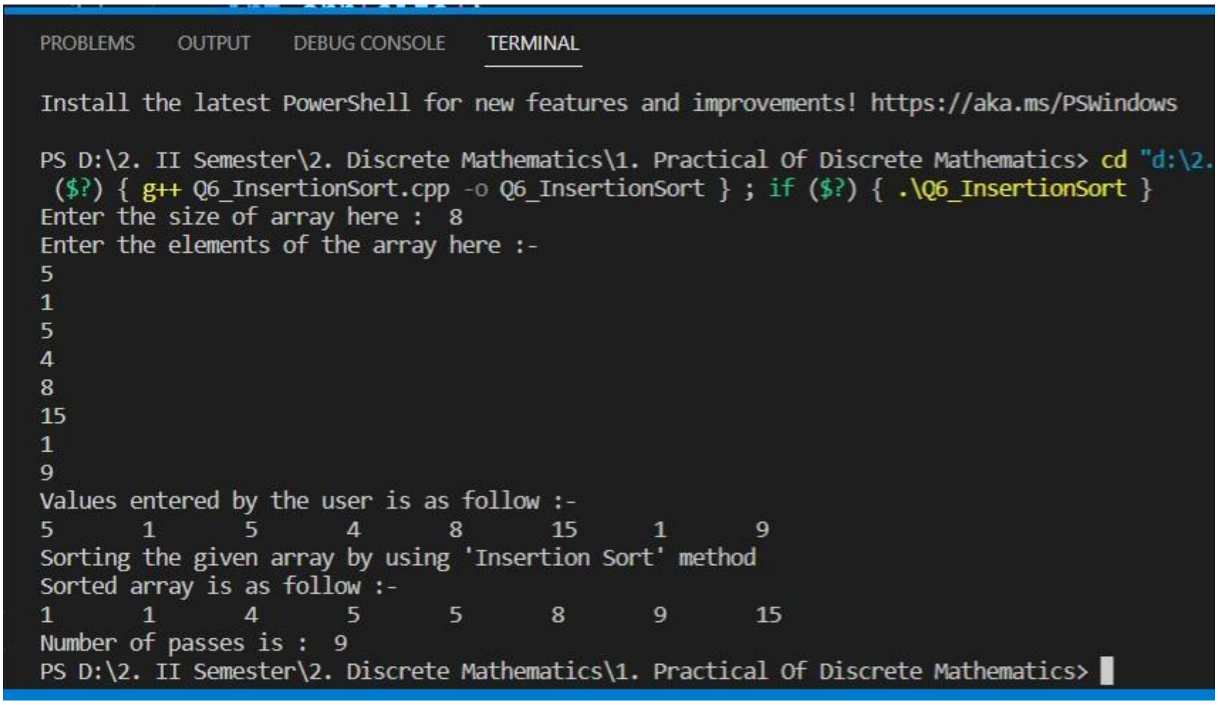
**->**

```
#include <iostream>

using namespace std;


int main()

{

    int size, ctr = 0;

    cout << "Enter the size of array here :  ";

    cin >> size;

    int arr[size];

    cout << "Enter the elements of the array here :-\n";

    for (int i = 0; i < size; i++)

    {

        cin >> arr[i];

    }

    cout << "Values entered by the user is as follow :-\n";

    for (int i = 0; i < size; i++)

    {

        cout << arr[i] << "\t";

    }

    cout << "\nSorting the given array by using 'Insertion Sort' method\n";

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

    {

        int current = arr[i];

        int j = i - 1;

        while (arr[j] > current && j >= 0)

        {

            arr[j + 1] = arr[j];

            j--;

            ctr++;

        }

        arr[j + 1] = current;

    }
```



```
    cout << "Sorted array is as follow :-\n";
    for (int i = 0; i < size; i++)
    {
        cout << arr[i] << "\t";
    }
    cout << "\nNumber of passes is :  " << ctr;

    return 0;
}
```

## Output Of 6th Program in IDE :-



```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd "d:\2.
 ($?) { g++ Q6_InsertionSort.cpp -o Q6_InsertionSort } ; if ($?) { .\Q6_InsertionSort }
Enter the size of array here :  8
Enter the elements of the array here :-
5
1
5
4
8
15
1
9
Values entered by the user is as follow :-
5       1       5       4       8       15      1       9
Sorting the given array by using 'Insertion Sort' method
Sorted array is as follow :-
1       1       4       5       5       8       9       15
Number of passes is :  9
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> 
```

**7. Write a Program that generates all the permutations of a given set of digits, with or without repetition.**

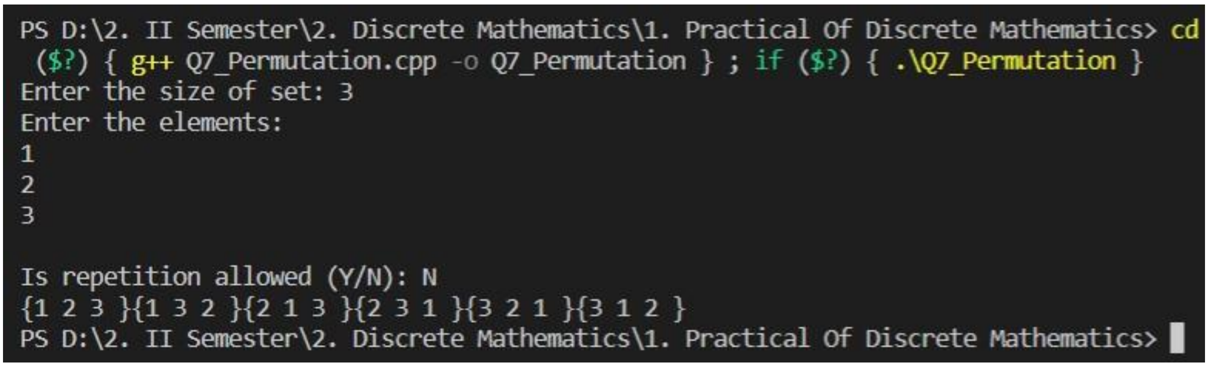
**(For example, if the given set is {1,2}, the permutations are 12 and 21). (One method is given in Liu)**

->

```
#include <iostream>
#define MAX_DIM 100

using namespace std;

void withRepetition(int *, int);
void withoutRepetition(int *, int);
void printWithRepetition(int *, int, int *, int, int);
void printWithoutRepetition(int *, int, int, int);
void swap(int &, int &);

void withRepetition(int *array, int size)
{
    int data[MAX_DIM] = {0};
    printWithRepetition(array, size, data, size - 1, 0);
    cout << endl;
}

void printWithRepetition(int *array, int size, int *data, int last, int index)
{
    for (int i = 0; i < size; i++)
    {
        data[index] = array[i];
        if (index == last)
        {
            cout << "{";
            for (int j = 0; j < index + 1; j++)
                cout << data[j] << " ";
            cout << "}";
        }

        else
        {
            printWithRepetition(array, size, data, last, index + 1);
        }
    }
}

void withoutRepetition(int *array, int size)
{
    printWithoutRepetition(array, size, 0, size - 1);
    cout << endl;
}

void printWithoutRepetition(int *array, int size, int start, int end)
{
    if (start == end)
    {
        cout << "{";
        for (int i = 0; i < size; i++)
            cout << array[i] << " ";
        cout << "}";
    }

    else
    {
        for (int i = start; i < end + 1; i++)
        {
            swap(array[start], array[i]);
            printWithoutRepetition(array, size, start + 1, end);
            swap(array[start], array[i]);
        }
    }
}

void swap(int &a, int &b)
{
    int t = b;
    b = a;
```

```cpp
    a = t;
}

int main()
{
    int size;
    char ch;

    cout << "Enter the size of set: ";
    cin >> size;

    int array[MAX_DIM];
    cout << "Enter the elements: ";
    for (int i = 0; i < size; i++)
        cin >> array[i];

    cout << "\nIs repetition allowed (Y/N): ";
    cin >> ch;

    switch (ch)
    {
    case 'Y':
        withRepetition(array, size);
        break;
    case 'N':
        withoutRepetition(array, size);
        break;
    default:
        cout << "\nWrong Choice";
    }

    return 0;
}
```

## Output Of 7th Program in IDE :-

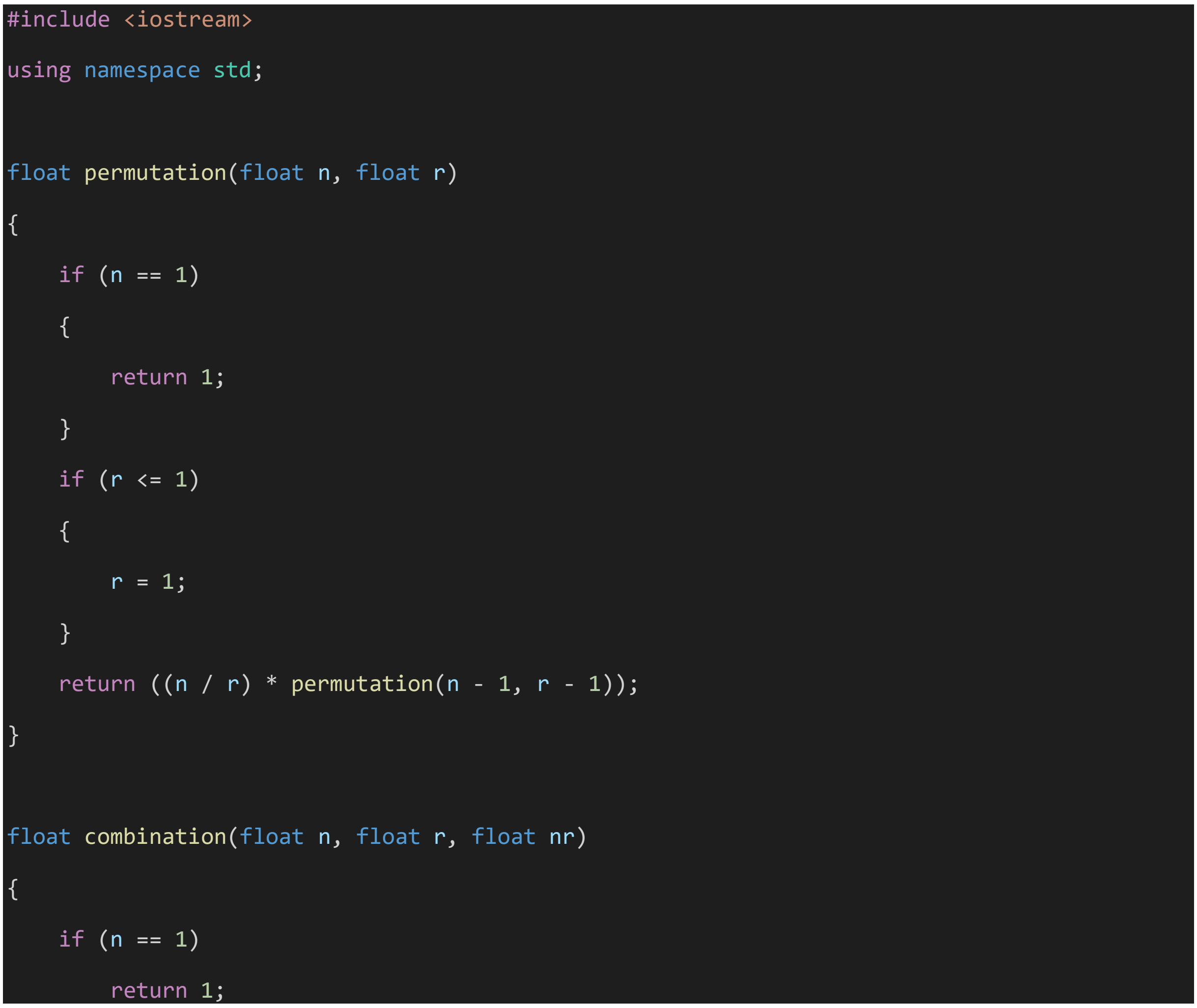


```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd
 ($?) { g++ Q7_Permutation.cpp -o Q7_Permutation } ; if ($?) { .\Q7_Permutation }
Enter the size of set: 3
Enter the elements:
1
2
3

Is repetition allowed (Y/N): N
{1 2 3 }{1 3 2 }{2 1 3 }{2 3 1 }{3 2 1 }{3 1 2 }
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> ▌
```

---

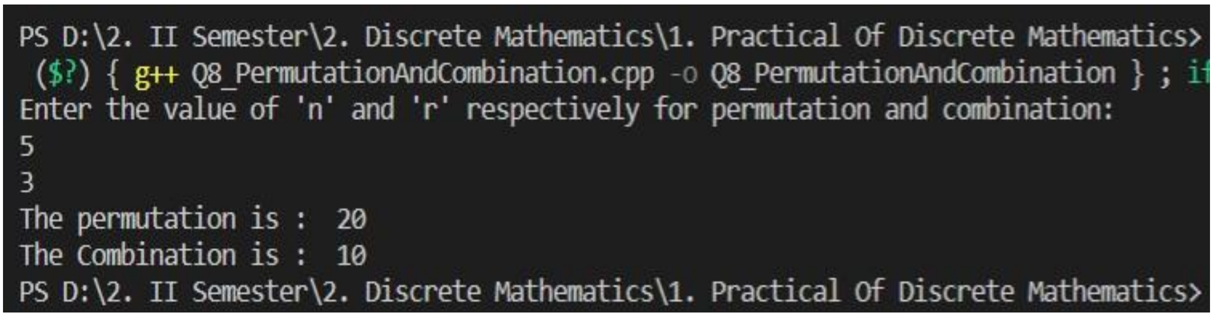
**8. Write a Program to calculate Permutation and Combination for an input value n and r using recursive formula of nCr and n Pr .**

**->**



```
#include <iostream>

using namespace std;


float permutation(float n, float r)

{

    if (n == 1)

    {

        return 1;

    }

    if (r <= 1)

    {

        r = 1;

    }

    return ((n / r) * permutation(n - 1, r - 1));

}


float combination(float n, float r, float nr)

{

    if (n == 1)

        return 1;
    if (r <= 1)
        r = 1;
    if (nr <= 1)
        nr = 1;

    return (n / (nr * r) * combination(n - 1, r - 1, nr - 1));
}

int main()
{
    int n, r;
    cout << "Enter the value of 'n' and 'r' respectively for permutation and
combination:\n";
    cin >> n >> r;
    while (r > n)
    {
        cout << "'r' should me less than or equal to n\nPlease re-enter the value of
'r'\n";
        cin >> r;
    }
    cout << "The permutation is : " << permutation(n, r);
    cout << endl
         << "The Combination is :  " << combination(n, r, (n - r));

    return 0;
}
```

## Output Of 8th Program in IDE :-



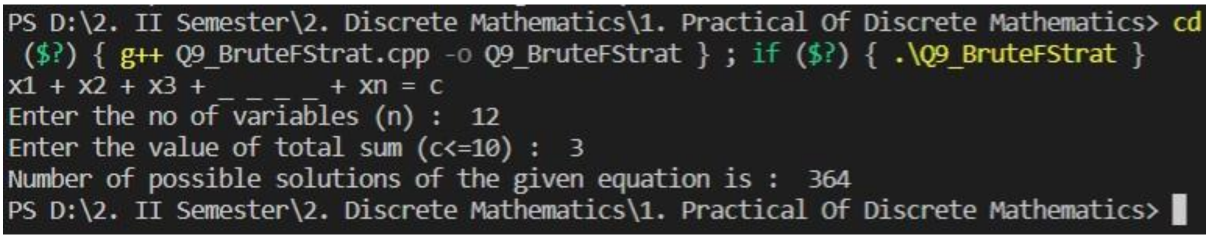
```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics>
 ($?) { g++ Q8_PermutationAndCombination.cpp -o Q8_PermutationAndCombination } ; if
Enter the value of 'n' and 'r' respectively for permutation and combination:
5
3
The permutation is :  20
The Combination is :  10
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics>
```

**9. For any number n, write a program to list all the solutions of the equation x1 + x2 + x3 + …+ xn = C, where C is a constant (C<=10) and x1, x2,x3,…,xn are nonnegative integers using brute force strategy.**

**->**

```
#include <iostream>
using namespace std;
int comb(int n, int r)
{
    if (r == 0 || r == n)
        return 1;
    else
        return (comb(n - 1, r - 1) + comb(n - 1, r));
}
int main()
{
    int n, tsum;
    cout << "x1 + x2 + x3 + _ _ _ _ + xn = c" << endl;
    cout << "Enter the no of variables (n) :  ";
    cin >> n;
    cout << "Enter the value of total sum (c<=10) : ";
    cin >> tsum;
    cout << "Number of possible solutions of the given equation is : ";
    cout << comb((n + tsum - 1), tsum);

    return 0;
}
```

### Output Of 9th Program in IDE :-



```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd
 ($?) { g++ Q9_BruteFStrat.cpp -o Q9_BruteFStrat } ; if ($?) { .\Q9_BruteFStrat }
x1 + x2 + x3 + _ _ _ _ + xn = c
Enter the no of variables (n) :  12
Enter the value of total sum (c<=10) :  3
Number of possible solutions of the given equation is :  364
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> █
```

**10.** Write a Program to accept the truth values of variables x and y, and print the truth table of the following logical operations:

a) Conjunction

b) Disjunction

c) Exclusive OR

d) Conditional

e) Bi-conditional

f) Exclusive NOR

g) Negation

h) NAND

i) NOR

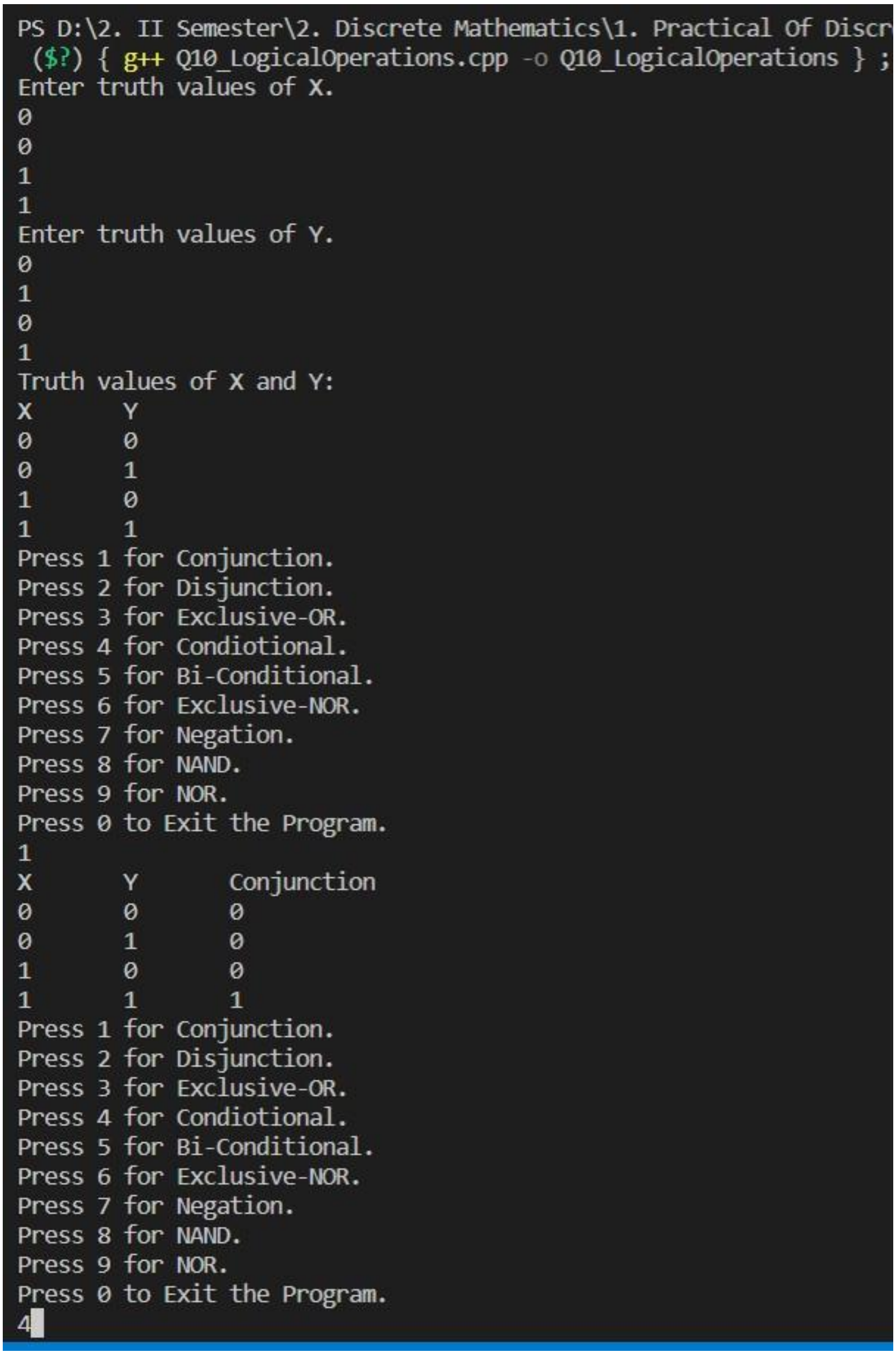
**->**

```
#include <iostream>
using namespace std;
int main()
{
    bool x[4], y[4], ans[4];
    int ch;
    cout << "Enter truth values of X." << endl;
    for (int i = 0; i < 4; i++)
    {
        cin >> x[i];
    }
    cout << "Enter truth values of Y." << endl;
    for (int i = 0; i < 4; i++)
    {
        cin >> y[i];
    }
    cout << "Truth values of X and Y:" << endl;
    cout << "X\tY" << endl;
    for (int i = 0; i < 4; i++)
    {
        cout << x[i] << "\t" << y[i] << endl;
    }
    do
    {
        cout << "Press 1 for Conjunction." << endl;
        cout << "Press 2 for Disjunction." << endl;
        cout << "Press 3 for Exclusive-OR." << endl;
        cout << "Press 4 for Condiotional." << endl;
        cout << "Press 5 for Bi-Conditional." << endl;
        cout << "Press 6 for Exclusive-NOR." << endl;
        cout << "Press 7 for Negation." << endl;
        cout << "Press 8 for NAND." << endl;
        cout << "Press 9 for NOR." << endl;
        cout << "Press 0 to Exit the Program.\n";
        cin >> ch;
        switch (ch)
        {
        case 1:
            for (int i = 0; i < 4; i++)
            {
                if (x[i] == 1 && y[i] == 1)
                {
                    ans[i] = 1;
                }
                else
                {
                    ans[i] = 0;
                }
            }
            cout << "X\tY\tConjunction" << endl;
            for (int i = 0; i < 4; i++)
            {
                cout << x[i] << "\t" << y[i] << "\t" << ans[i] << endl;
            }
            break;

        case 2:
            for (int i = 0; i < 4; i++)
            {
                if (x[i] == 0 && y[i] == 0)
                {
                    ans[i] = 0;
                }
                else
                {
                    ans[i] = 1;
                }
            }
            cout << "X\tY\tDisjunction" << endl;
            for (int i = 0; i < 4; i++)
            {
                cout << x[i] << "\t" << y[i] << "\t" << ans[i] << endl;
            }
            break;

        case 3:
            for (int i = 0; i < 4; i++)
            {
                if (x[i] == y[i])
                {
                    ans[i] = 0;
                }
                else
                {
                    ans[i] = 1;
                }
            }
            cout << "X\tY\tExclusive-OR" << endl;
            for (int i = 0; i < 4; i++)
            {
                cout << x[i] << "\t" << y[i] << "\t" << ans[i] << endl;
            }
            break;

        case 4:
            for (int i = 0; i < 4; i++)
            {
                if (x[i] == 1 && y[i] == 0)
                {
                    ans[i] = 0;
                }
                else
                {
                    ans[i] = 1;
                }
            }
            cout << "X\tY\tConditional" << endl;
            for (int i = 0; i < 4; i++)
            {
                cout << x[i] << "\t" << y[i] << "\t" << ans[i] << endl;
            }
            break;

        case 5:
            for (int i = 0; i < 4; i++)
            {
                if (x[i] == y[i])
                {
                    ans[i] = 1;
                }
                else
                {
                    ans[i] = 0;
                }
            }
            cout << "X\tY\tBi-Conditional" << endl;
            for (int i = 0; i < 4; i++)
            {
                cout << x[i] << "\t" << y[i] << "\t" << ans[i] << endl;
            }
            break;
```

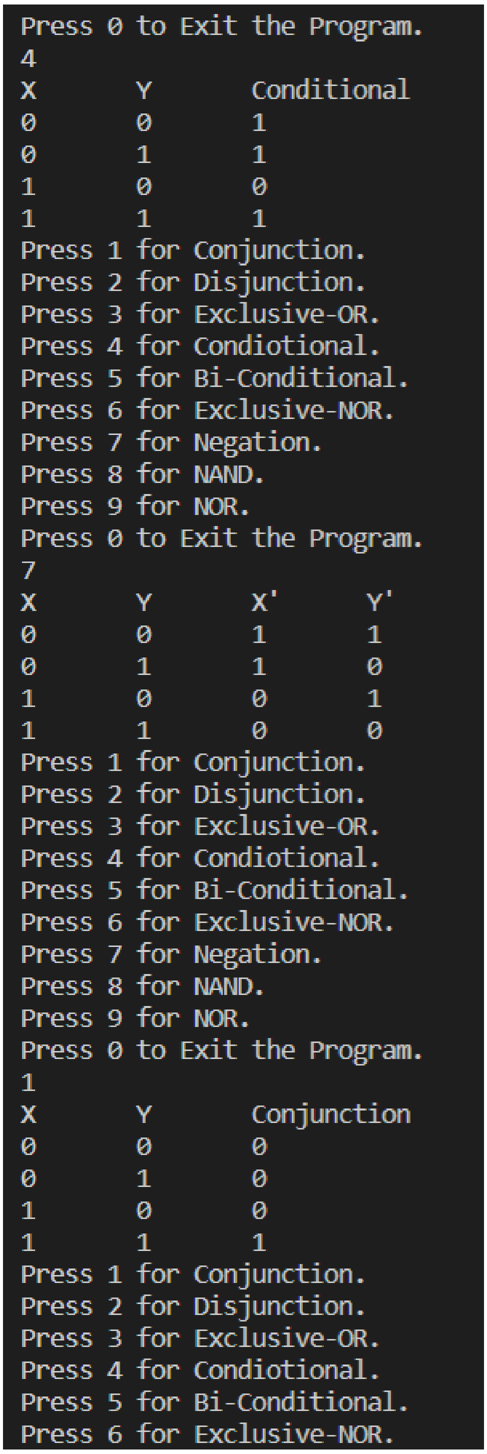
```
case 6:
    for (int i = 0; i < 4; i++)
    {
        if (x[i] == y[i])
        {
            ans[i] = 0;
        }
        else
        {
            ans[i] = 1;
        }
    }
    break;
    cout << "X\tY\tExclusive-NOR" << endl;
    for (int i = 0; i < 4; i++)
    {
        cout << x[i] << "\t" << y[i] << "\t" << !ans[i] << endl;
    }
    break;

case 7:
    cout << "X\tY\tX'\tY'" << endl;
    for (int i = 0; i < 4; i++)
    {
        cout << x[i] << "\t" << y[i] << "\t" << !x[i] << "\t" << !y[i] << endl;
    }
    break;

case 8:
    for (int i = 0; i < 4; i++)
    {
        if (x[i] == 1 && y[i] == 1)
        {
            ans[i] = 1;
```

```
                }
                else
                {
                    ans[i] = 0;
                }
            }
            cout << "X\tY\tNAND" << endl;
            for (int i = 0; i < 4; i++)
            {
                cout << x[i] << "\t" << y[i] << "\t" << !ans[i] << endl;
            }
            break;

        case 9:
            for (int i = 0; i < 4; i++)
            {
                if (x[i] == 0 && y[i] == 0)
                {
                    ans[i] = 0;
                }
                else
                {
                    ans[i] = 1;
                }
            }
            cout << "X\tY\tNOR" << endl;
            for (int i = 0; i < 4; i++)
            {
                cout << x[i] << "\t" << y[i] << "\t" << !ans[i] << endl;
            }
            break;
        }
    } while (ch != 0);
    return 0;
}
```

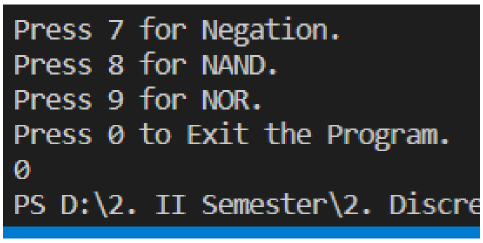
## Output Of 10th Program in IDE :-



```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discr
 ($?) { g++ Q10_LogicalOperations.cpp -o Q10_LogicalOperations } ;
Enter truth values of X.
0
0
1
1
Enter truth values of Y.
0
1
0
1
Truth values of X and Y:
X       Y
0       0
0       1
1       0
1       1
Press 1 for Conjunction.
Press 2 for Disjunction.
Press 3 for Exclusive-OR.
Press 4 for Condiotional.
Press 5 for Bi-Conditional.
Press 6 for Exclusive-NOR.
Press 7 for Negation.
Press 8 for NAND.
Press 9 for NOR.
Press 0 to Exit the Program.
1
X       Y       Conjunction
0       0       0
0       1       0
1       0       0
1       1       1
Press 1 for Conjunction.
Press 2 for Disjunction.
Press 3 for Exclusive-OR.
Press 4 for Condiotional.
Press 5 for Bi-Conditional.
Press 6 for Exclusive-NOR.
Press 7 for Negation.
Press 8 for NAND.
Press 9 for NOR.
Press 0 to Exit the Program.
4
```



```
Press 0 to Exit the Program.
4
X       Y       Conditional
0       0       1
0       1       1
1       0       0
1       1       1
Press 1 for Conjunction.
Press 2 for Disjunction.
Press 3 for Exclusive-OR.
Press 4 for Condiotional.
Press 5 for Bi-Conditional.
Press 6 for Exclusive-NOR.
Press 7 for Negation.
Press 8 for NAND.
Press 9 for NOR.
Press 0 to Exit the Program.
7
X       Y       X'      Y'
0       0       1       1
0       1       1       0
1       0       0       1
1       1       0       0
Press 1 for Conjunction.
Press 2 for Disjunction.
Press 3 for Exclusive-OR.
Press 4 for Condiotional.
Press 5 for Bi-Conditional.
Press 6 for Exclusive-NOR.
Press 7 for Negation.
Press 8 for NAND.
Press 9 for NOR.
Press 0 to Exit the Program.
1
X       Y       Conjunction
0       0       0
0       1       0
1       0       0
1       1       1
Press 1 for Conjunction.
Press 2 for Disjunction.
Press 3 for Exclusive-OR.
Press 4 for Condiotional.
Press 5 for Bi-Conditional.
Press 6 for Exclusive-NOR.
```



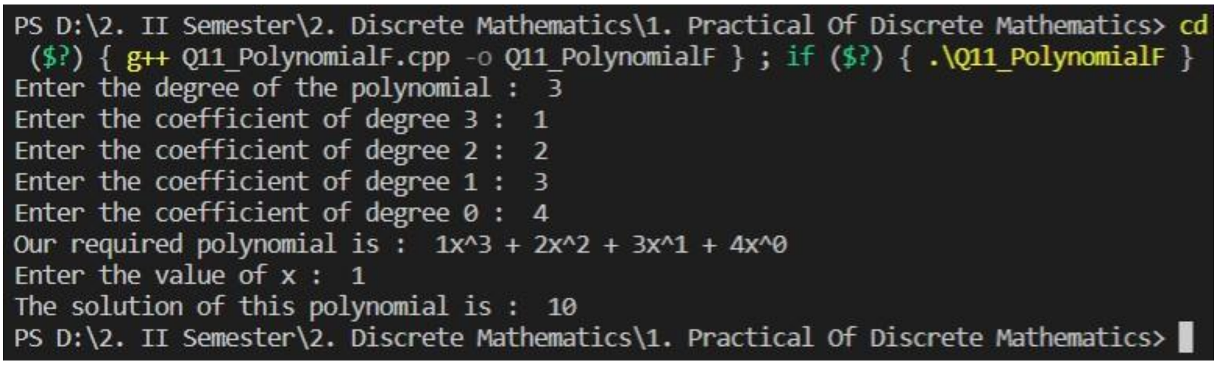
```
Press 7 for Negation.
Press 8 for NAND.
Press 9 for NOR.
Press 0 to Exit the Program.
0
PS D:\2. II Semester\2. Discre
```

**11. Write a Program to store a function (polynomial/exponential), and then evaluate the polynomial. (For example store f(x) = 4n3 + 2n + 9 in an array and for a given value of n, say n = 5, evaluate (i.e. compute the value of f(5)).**

**->**

```
#include <iostream>
#include <math.h>
using namespace std;
int main(){
    int arr[25], deg, x, sum = 0;
    char ch;
    cout << "Enter the degree of the polynomial : ";
    cin >> deg;
    for (int i = deg; i >= 0; i--)
    {
        cout << "Enter the coefficient of degree " << i << " : ";
        cin >> arr[i];
    }
    cout << "Our required polynomial is : ";
    cout << arr[deg] << "x^" << deg;
    for (int i = deg - 1; i > 0; i--)
    {
        if (arr[i] > 0)
            cout << " + " << arr[i] << "x^" << i;
        else
            cout << " - " << arr[i] << "x^" << i;
    }
    cout << " + " << arr[0] << "x^0" << endl;
    cout << "Enter the value of x :  ";
    cin >> x;
    for (int i = deg; i >= 0; i--)
    {
        sum += (arr[i] * pow(x, i));
    }
    cout << "The solution of this polynomial is :  " << sum;
}
```

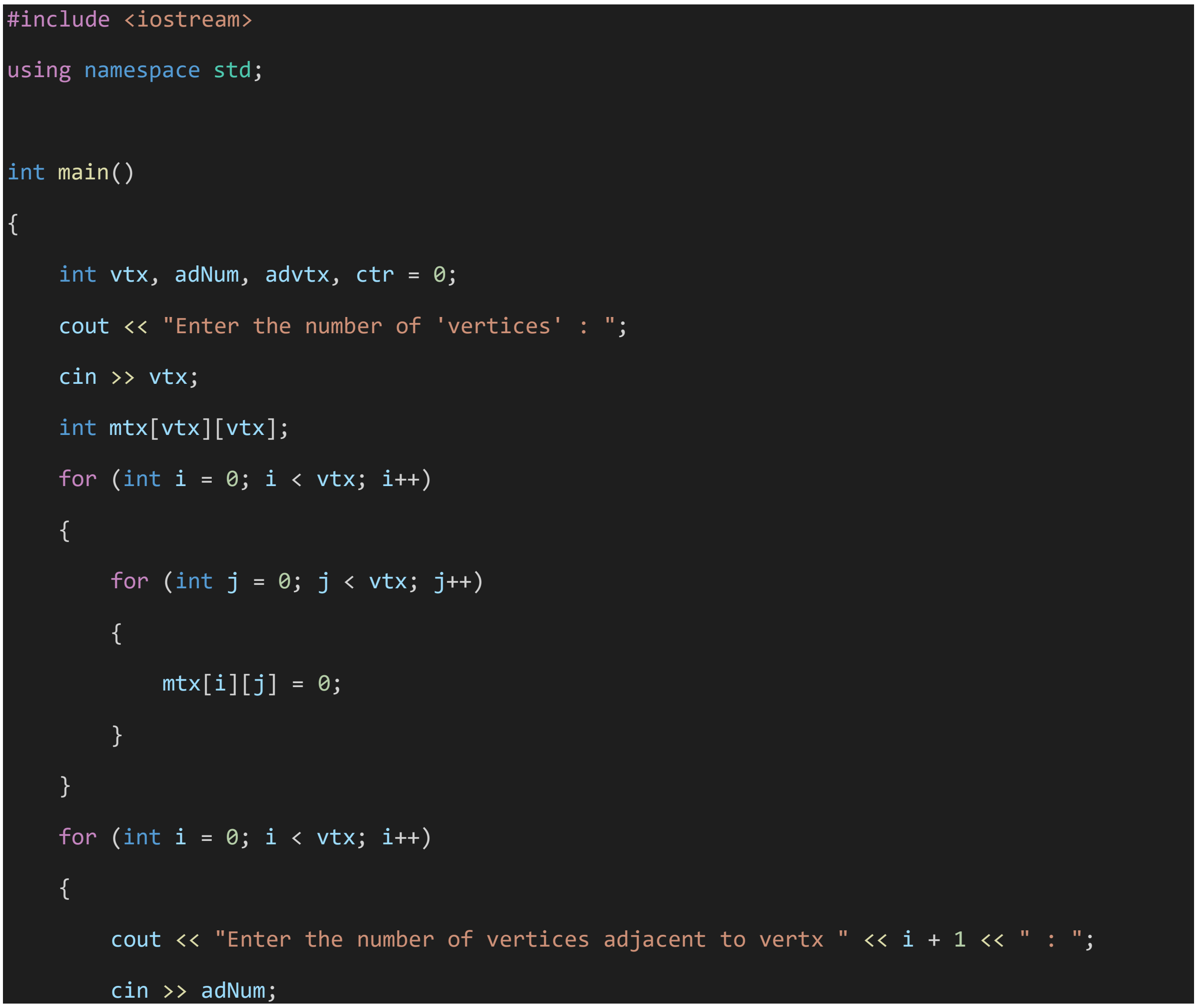
## Output Of $11^{th}$ Program in IDE :-



```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd
 ($?) { g++ Q11_PolynomialF.cpp -o Q11_PolynomialF } ; if ($?) { .\Q11_PolynomialF }
Enter the degree of the polynomial :  3
Enter the coefficient of degree 3 :  1
Enter the coefficient of degree 2 :  2
Enter the coefficient of degree 1 :  3
Enter the coefficient of degree 0 :  4
Our required polynomial is :  1x^3 + 2x^2 + 3x^1 + 4x^0
Enter the value of x :  1
The solution of this polynomial is :  10
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> █
```

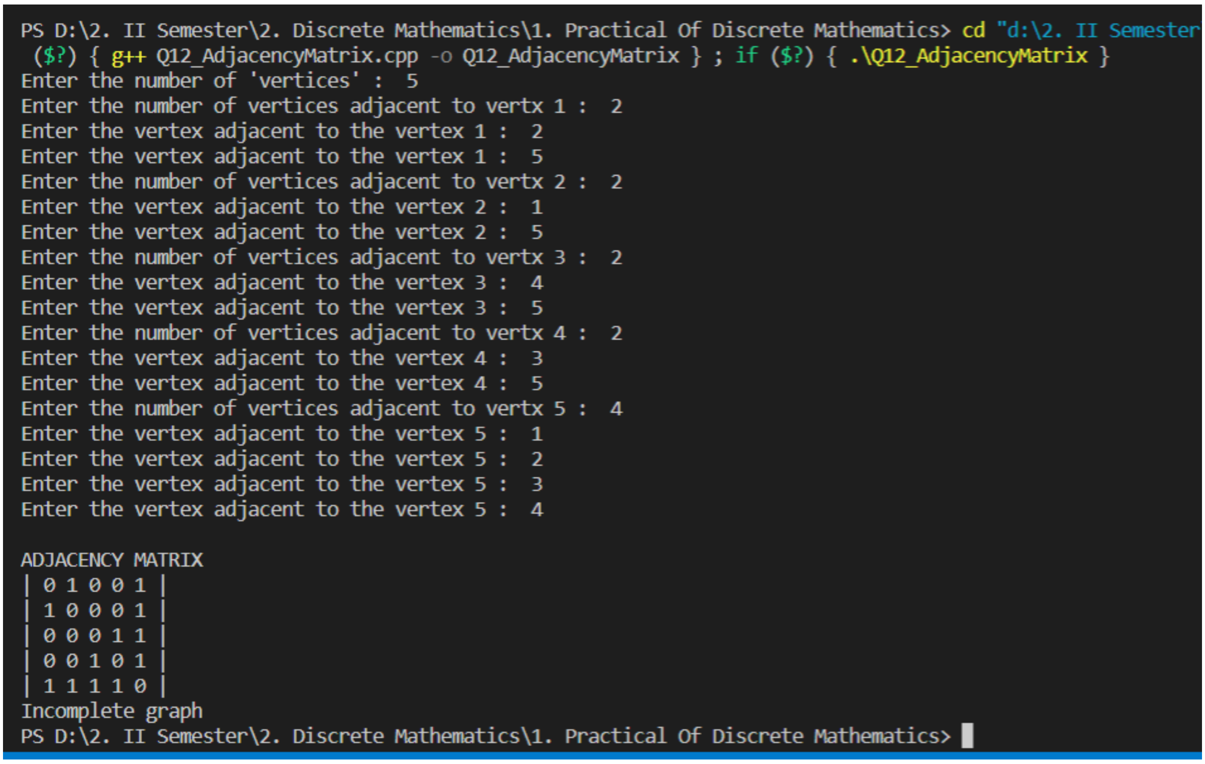
**12. Write a Program to represent Graphs using the Adjacency Matrices and check if it is a complete graph.**

**->**



```
#include <iostream>

using namespace std;


int main()

{

    int vtx, adNum, advtx, ctr = 0;

    cout << "Enter the number of 'vertices' : ";

    cin >> vtx;

    int mtx[vtx][vtx];

    for (int i = 0; i < vtx; i++)

    {

        for (int j = 0; j < vtx; j++)

        {

            mtx[i][j] = 0;

        }

    }

    for (int i = 0; i < vtx; i++)

    {

        cout << "Enter the number of vertices adjacent to vertx " << i + 1 << " : ";

        cin >> adNum;
        for (int j = 0; j < adNum; j++)
        {
            cout << "Enter the vertex adjacent to the vertex " << i + 1 << " :  ";
            cin >> advtx;
            mtx[i][advtx - 1] = 1;
        }
    }
    cout << "\nADJACENCY MATRIX\n";
    for (int i = 0; i < vtx; i++)
    {
        int sum = 0;
        cout << "| ";
        for (int j = 0; j < vtx; j++)
        {
            cout << mtx[i][j] << " ";
            if (mtx[i][i] == 0)
                sum += mtx[i][j];
        }
        cout << "|";
        cout << endl;
        if (sum == (vtx - 1))
        {
            ctr++;
        }
    }
    if (ctr == vtx)
        cout << "Complete graph";
    else
        cout << "Incomplete graph";

    return 0;
}
```

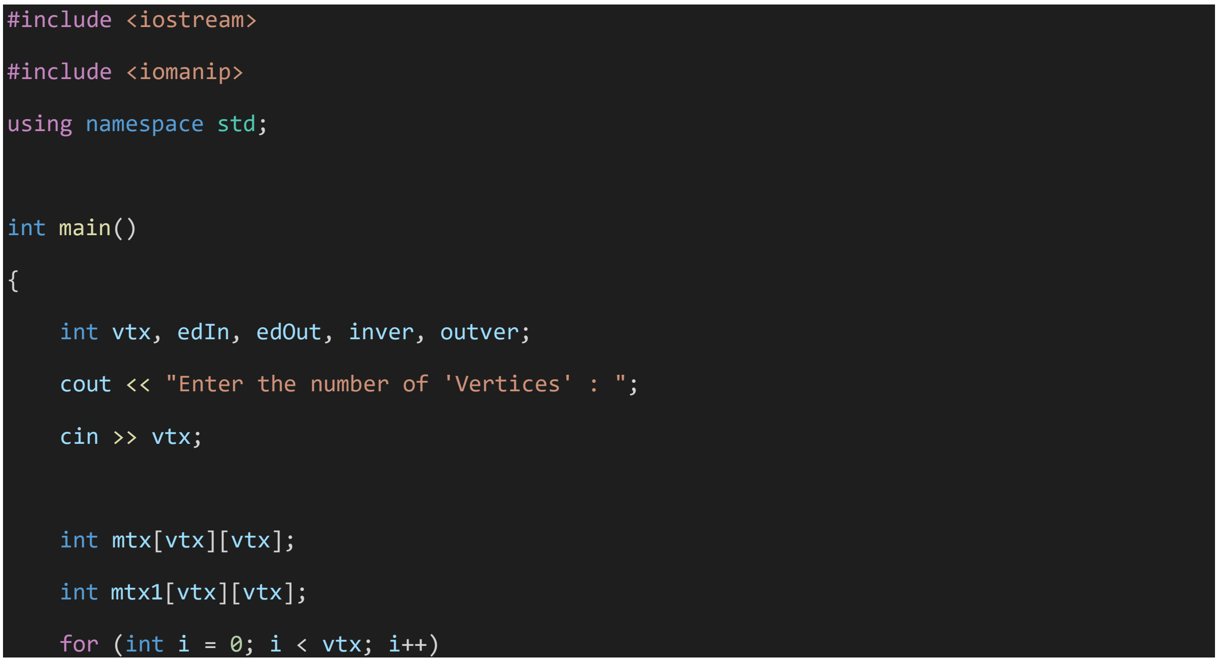
## Output Of 12th Program in IDE :-



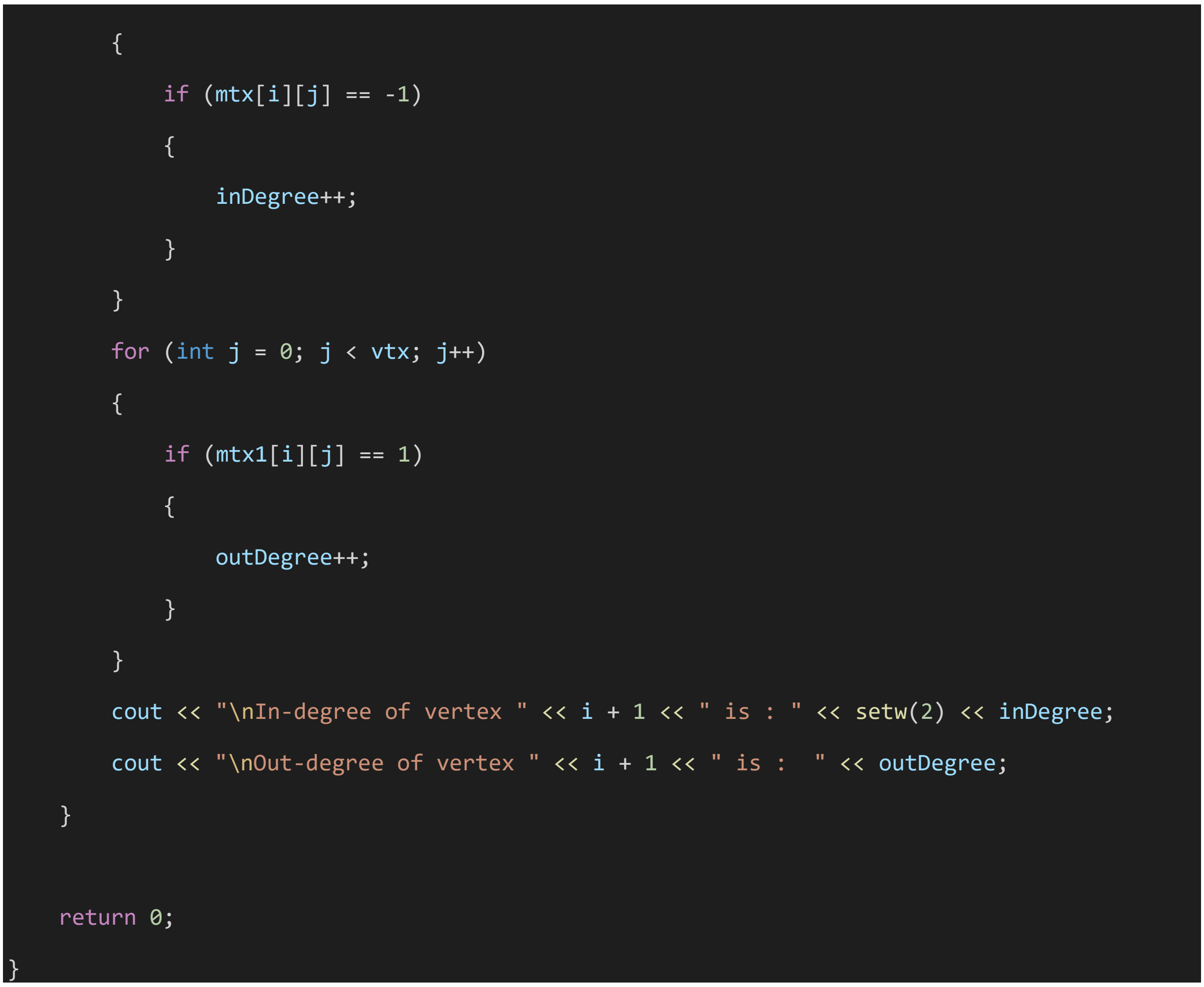
```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd "d:\2. II Semester
 ($?) { g++ Q12_AdjacencyMatrix.cpp -o Q12_AdjacencyMatrix } ; if ($?) { .\Q12_AdjacencyMatrix }
Enter the number of 'vertices' :  5
Enter the number of vertices adjacent to vertx 1 :  2
Enter the vertex adjacent to the vertex 1 :  2
Enter the vertex adjacent to the vertex 1 :  5
Enter the number of vertices adjacent to vertx 2 :  2
Enter the vertex adjacent to the vertex 2 :  1
Enter the vertex adjacent to the vertex 2 :  5
Enter the number of vertices adjacent to vertx 3 :  2
Enter the vertex adjacent to the vertex 3 :  4
Enter the vertex adjacent to the vertex 3 :  5
Enter the number of vertices adjacent to vertx 4 :  2
Enter the vertex adjacent to the vertex 4 :  3
Enter the vertex adjacent to the vertex 4 :  5
Enter the number of vertices adjacent to vertx 5 :  4
Enter the vertex adjacent to the vertex 5 :  1
Enter the vertex adjacent to the vertex 5 :  2
Enter the vertex adjacent to the vertex 5 :  3
Enter the vertex adjacent to the vertex 5 :  4

ADJACENCY MATRIX
| 0 1 0 0 1 |
| 1 0 0 0 1 |
| 0 0 0 1 1 |
| 0 0 1 0 1 |
| 1 1 1 1 0 |
Incomplete graph
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics>
```

**13. Write a Program to accept a directed graph G and compute the in-degree and out-degree of each vertex.**

**->**

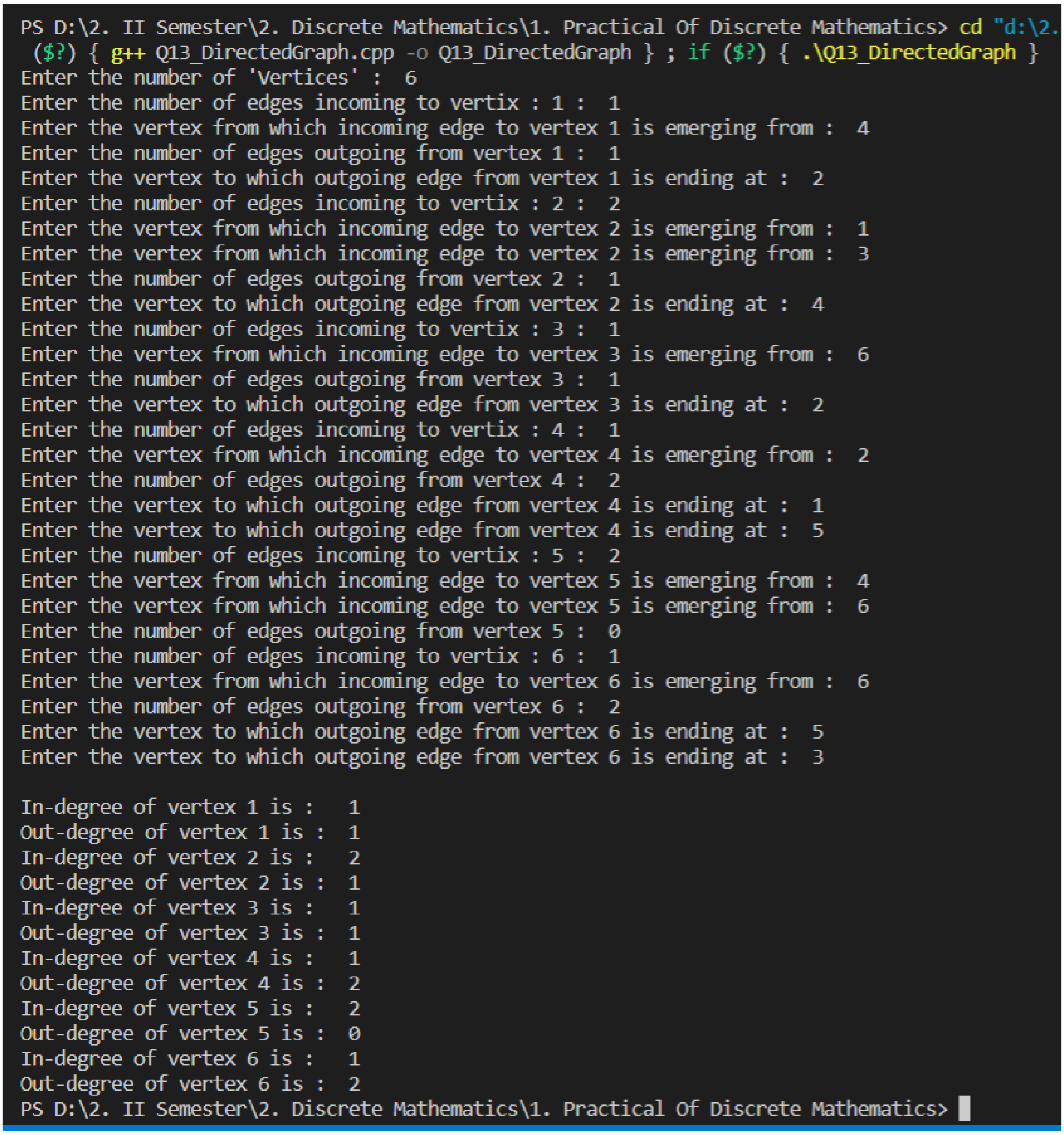


```
#include <iostream>

#include <iomanip>

using namespace std;


int main()

{

    int vtx, edIn, edOut, inver, outver;

    cout << "Enter the number of 'Vertices' : ";

    cin >> vtx;


    int mtx[vtx][vtx];

    int mtx1[vtx][vtx];

    for (int i = 0; i < vtx; i++)
    {
        for (int j = 0; j < vtx; j++)
        {
            mtx[i][j] = 0;
            mtx1[i][j] = 0;
        }
    }

    for (int i = 0; i < vtx; i++)
    {
        cout << "Enter the number of edges incoming to vertix : " << i + 1 << " :  ";
        cin >> edIn;
        for (int j = 0; j < edIn; j++)
        {
            cout << "Enter the vertex from which incoming edge to vertex " << i + 1 << " is emerging from :  ";
            cin >> inver;
            mtx[i][inver - 1] = -1;
        }

        cout << "Enter the number of edges outgoing from vertex " << i + 1 << " :  ";
        cin >> edOut;
        for (int k = 0; k < edOut; k++)
        {
            cout << "Enter the vertex to which outgoing edge from vertex " << i + 1 << " is ending at :  ";
            cin >> outver;
            mtx1[i][outver - 1] = 1;
        }
    }

    for (int i = 0; i < vtx; i++)
    {
        int inDegree = 0, outDegree = 0;
        for (int j = 0; j < vtx; j++)
```



```
        {
            if (mtx[i][j] == -1)
            {
                inDegree++;
            }
        }
        for (int j = 0; j < vtx; j++)
        {
            if (mtx1[i][j] == 1)
            {
                outDegree++;
            }
        }
        cout << "\nIn-degree of vertex " << i + 1 << " is : " << setw(2) << inDegree;
        cout << "\nOut-degree of vertex " << i + 1 << " is :  " << outDegree;
    }

    return 0;
}
```

## Output Of 13th Program in IDE :-

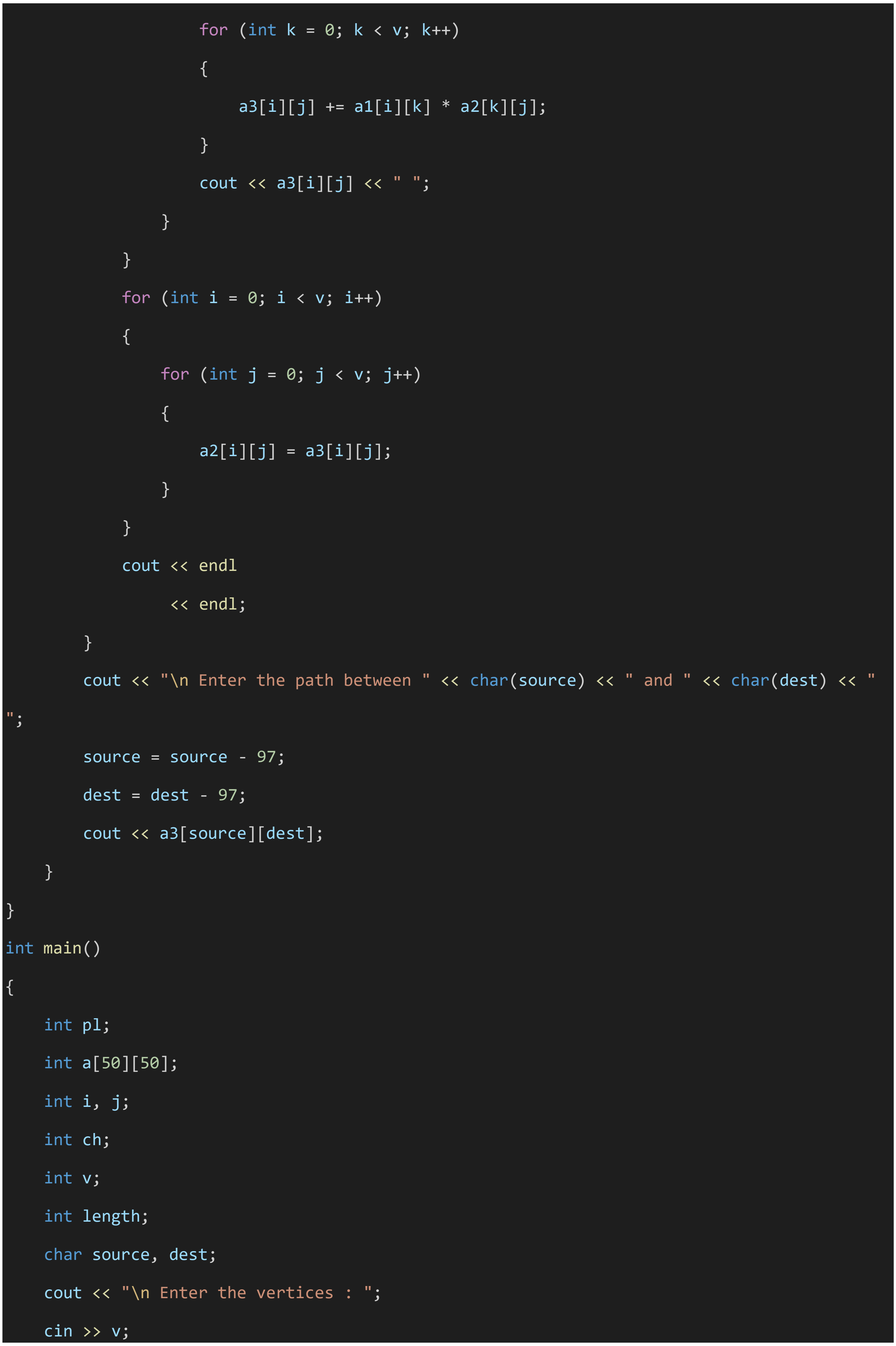


```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd "d:\2.
 ($?) { g++ Q13_DirectedGraph.cpp -o Q13_DirectedGraph } ; if ($?) { .\Q13_DirectedGraph }
Enter the number of 'Vertices' :  6
Enter the number of edges incoming to vertix : 1 :  1
Enter the vertex from which incoming edge to vertex 1 is emerging from :  4
Enter the number of edges outgoing from vertex 1 :  1
Enter the vertex to which outgoing edge from vertex 1 is ending at :  2
Enter the number of edges incoming to vertix : 2 :  2
Enter the vertex from which incoming edge to vertex 2 is emerging from :  1
Enter the vertex from which incoming edge to vertex 2 is emerging from :  3
Enter the number of edges outgoing from vertex 2 :  1
Enter the vertex to which outgoing edge from vertex 2 is ending at :  4
Enter the number of edges incoming to vertix : 3 :  1
Enter the vertex from which incoming edge to vertex 3 is emerging from :  6
Enter the number of edges outgoing from vertex 3 :  1
Enter the vertex to which outgoing edge from vertex 3 is ending at :  2
Enter the number of edges incoming to vertix : 4 :  1
Enter the vertex from which incoming edge to vertex 4 is emerging from :  2
Enter the number of edges outgoing from vertex 4 :  2
Enter the vertex to which outgoing edge from vertex 4 is ending at :  1
Enter the vertex to which outgoing edge from vertex 4 is ending at :  5
Enter the number of edges incoming to vertix : 5 :  2
Enter the vertex from which incoming edge to vertex 5 is emerging from :  4
Enter the vertex from which incoming edge to vertex 5 is emerging from :  6
Enter the number of edges outgoing from vertex 5 :  0
Enter the number of edges incoming to vertix : 6 :  1
Enter the vertex from which incoming edge to vertex 6 is emerging from :  6
Enter the number of edges outgoing from vertex 6 :  2
Enter the vertex to which outgoing edge from vertex 6 is ending at :  5
Enter the vertex to which outgoing edge from vertex 6 is ending at :  3

In-degree of vertex 1 is :   1
Out-degree of vertex 1 is :  1
In-degree of vertex 2 is :   2
Out-degree of vertex 2 is :  1
In-degree of vertex 3 is :   1
Out-degree of vertex 3 is :  1
In-degree of vertex 4 is :   1
Out-degree of vertex 4 is :  2
In-degree of vertex 5 is :   2
Out-degree of vertex 5 is :  0
In-degree of vertex 6 is :   1
Out-degree of vertex 6 is :  2
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> 
```

**14. Given a graph G, write a Program to find the number of paths of length n between the source and destination entered by the user.**

->

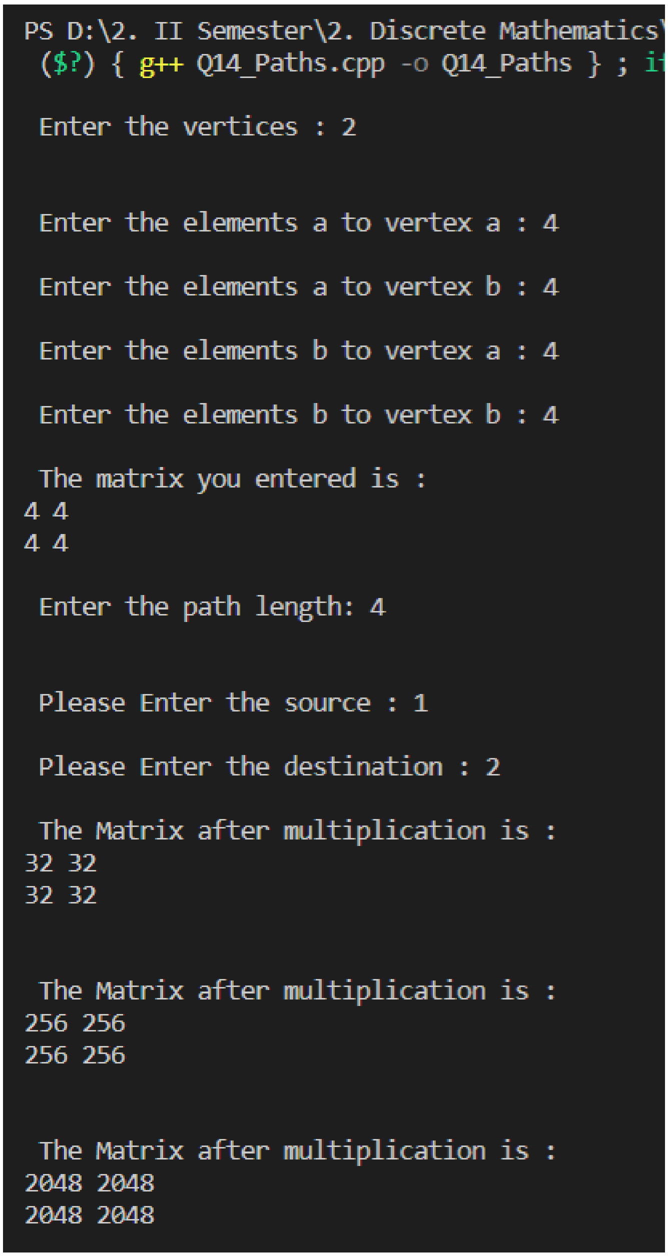
```cpp
#include <iostream>

using namespace std;

void multiplication(int a1[50][50], int v, int pl, int source, int dest)

{

    int a3[50][50], a2[50][50];

    for (int i = 0; i < v; i++)

    {

        for (int j = 0; j < v; j++)

        {

            a2[i][j] = a1[i][j];

        }

    }

    if (pl == 1)

    {

        for (int i = 0; i < v; i++)

        {

            for (int j = 0; j < v; j++)

            {

                cout << a1[i][j] << " ";

            }

            cout << endl;

        }

    }

    else

    {

        for (int l = 2; l <= pl; l++)

        {

            cout << "\n The Matrix after multiplication is : ";

            for (int i = 0; i < v; i++)

            {

                cout << endl;

                for (int j = 0; j < v; j++)

                {

                    a3[i][j] = 0;
```



```
                    for (int k = 0; k < v; k++)
                    {
                        a3[i][j] += a1[i][k] * a2[k][j];
                    }
                    cout << a3[i][j] << " ";
                }
            }
            for (int i = 0; i < v; i++)
            {
                for (int j = 0; j < v; j++)
                {
                    a2[i][j] = a3[i][j];
                }
            }
            cout << endl
                 << endl;
        }
        cout << "\n Enter the path between " << char(source) << " and " << char(dest) << " 
";
        source = source - 97;
        dest = dest - 97;
        cout << a3[source][dest];
    }
}
int main()
{
    int pl;
    int a[50][50];
    int i, j;
    int ch;
    int v;
    int length;
    char source, dest;
    cout << "\n Enter the vertices : ";
    cin >> v;
```

```cpp
    cout << endl;
    for (int i = 0; i < v; i++)
    {
        for (int j = 0; j < v; j++)
        {
            cout << "\n Enter the elements ";
            cout << (char)(i + 97) << " "<< "to vertex"<< " " << (char)(j + 97) << " : ";
            cin >> a[i][j];
        }
    }
    cout << "\n The matrix you entered is : " << endl;
    for (i = 0; i < v; i++)
    {
        for (j = 0; j < v; j++)
        {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    cout << "\n Enter the path length: ";
    cin >> pl;
    cout << endl;
    cout << "\n Please Enter the source : ";
    cin >> source;
    cout << "\n Please Enter the destination : ";
    cin >> dest;
    multiplication(a, v, pl, source, dest);
    return 0;
}
```

## Output Of 14th Program in IDE :-



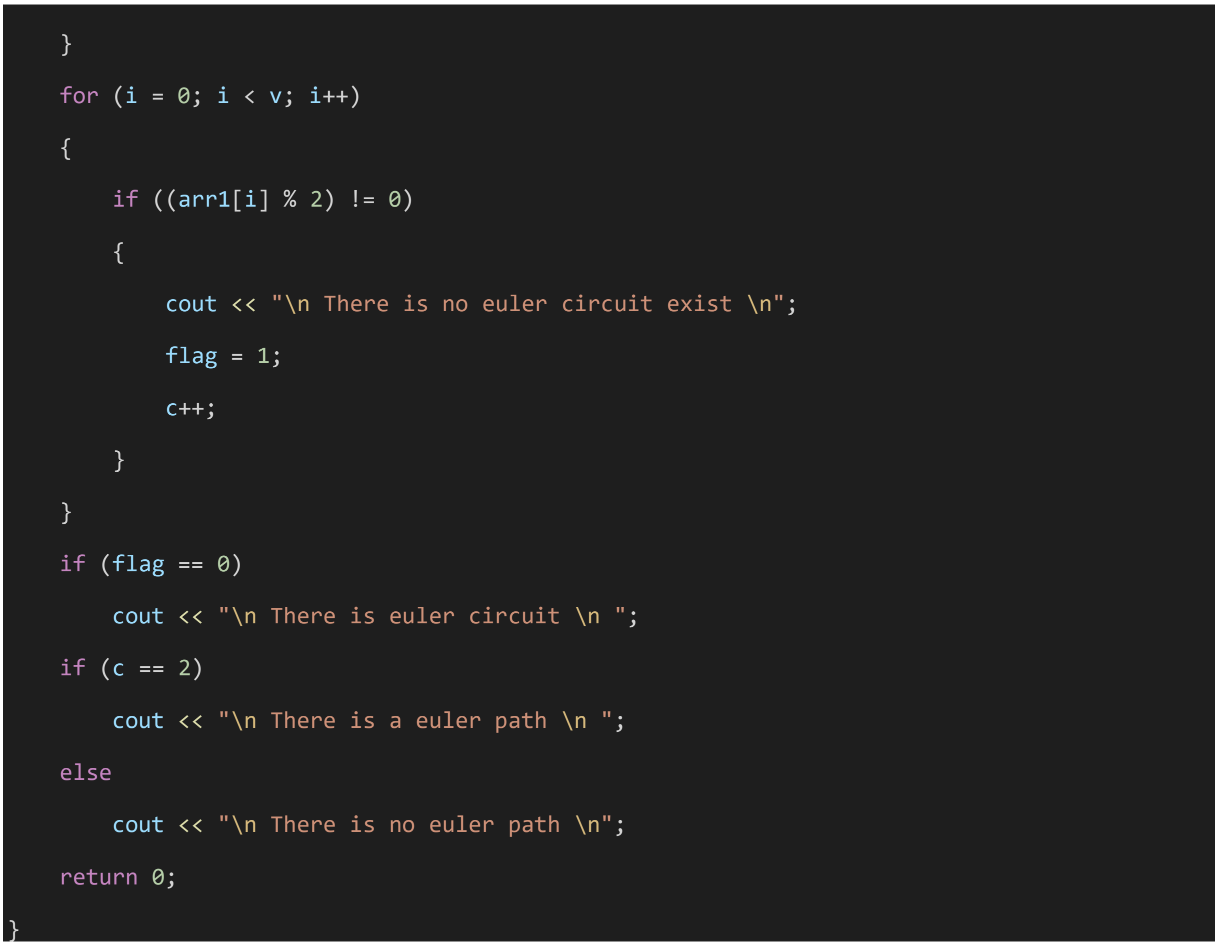
```
PS D:\2. II Semester\2. Discrete Mathematics\
 ($?) { g++ Q14_Paths.cpp -o Q14_Paths } ; i

 Enter the vertices : 2


 Enter the elements a to vertex a : 4

 Enter the elements a to vertex b : 4

 Enter the elements b to vertex a : 4

 Enter the elements b to vertex b : 4

 The matrix you entered is :
4 4
4 4

 Enter the path length: 4


 Please Enter the source : 1

 Please Enter the destination : 2

 The Matrix after multiplication is :
32 32
32 32


 The Matrix after multiplication is :
256 256
256 256


 The Matrix after multiplication is :
2048 2048
2048 2048
```

**15. Given an adjacency matrix of a graph, write a program to check whether a given set of vertices {v1,v2,v3. ...,vk} forms an Euler path / Euler Circuit (for circuit assume vk=v1).**

**->**

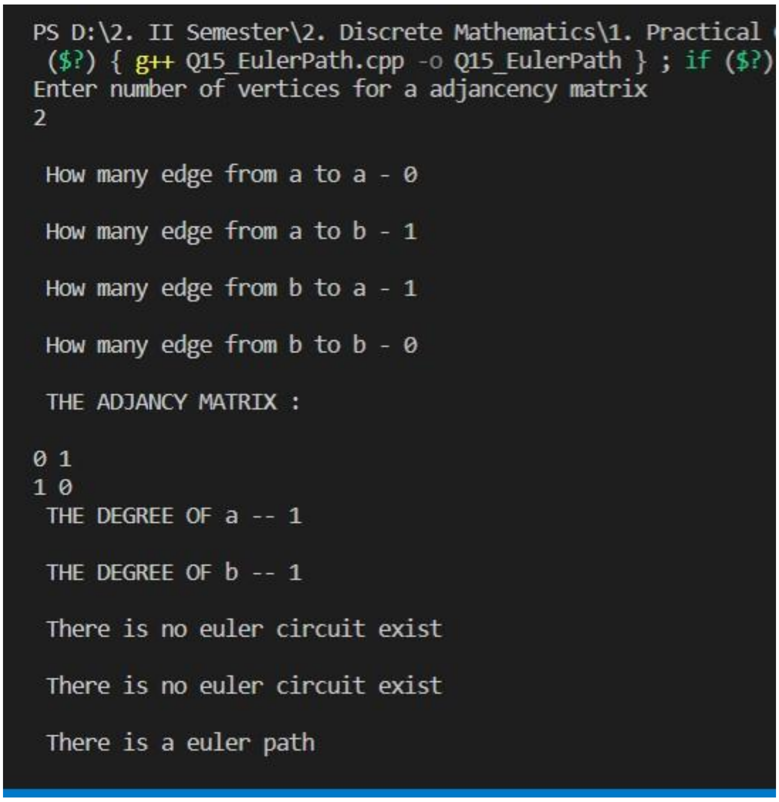
```
#include <iostream>

using namespace std;
```

```
int main()
{
    char charr[50], choice;
    int v, i, j, p = 0, sum = 0, flag = 0, c = 0;
    cout << "Enter number of vertices for a adjancency matrix \n";
    cin >> v;
    int arr[v][v], arr1[v];
    for (i = 0; i < v; i++)
    {
        for (j = 0; j < v; j++)
        {
            cout << "\n How many edge from " << (char)(97 + i) << " to " << (char)(97 + j)
<< " - ";
            cin >> arr[i][j];
        }
    }
    cout << "\n THE ADJANCY MATRIX : \n ";
    for (int m = 0; m < v; m++)
    {
        cout << endl;
        for (int n = 0; n < v; n++)
            cout << arr[m][n] << " ";
    }
    for (i = 0; i < v; i++)
    {
        sum = 0;
        for (j = 0; j < v; j++)
        {
            sum += arr[i][j];
        }
        arr1[i] = sum;
    }
    for (i = 0; i < v; i++)
    {
        cout << "\n THE DEGREE OF " << (char)(97 + i) << " -- " << arr1[i] << endl;
```



```
    }
    for (i = 0; i < v; i++)
    {
        if ((arr1[i] % 2) != 0)
        {
            cout << "\n There is no euler circuit exist \n";
            flag = 1;
            c++;
        }
    }
    if (flag == 0)
        cout << "\n There is euler circuit \n ";
    if (c == 2)
        cout << "\n There is a euler path \n ";
    else
        cout << "\n There is no euler path \n";
    return 0;
}
```

## Output Of 15th Program in IDE :-



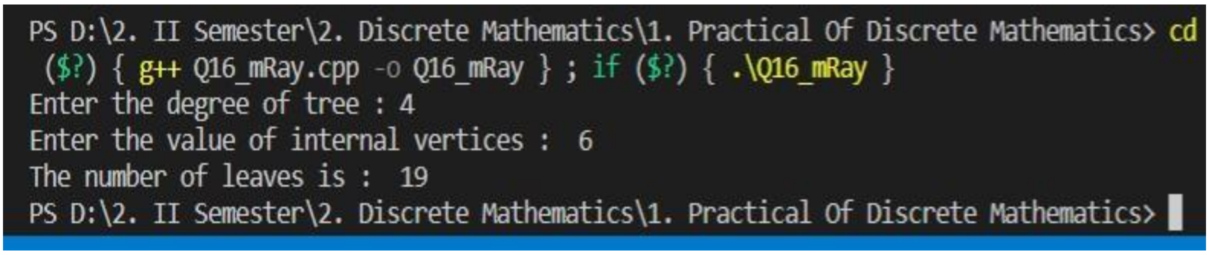
```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical
 ($?) { g++ Q15_EulerPath.cpp -o Q15_EulerPath } ; if ($?)
Enter number of vertices for a adjancency matrix
2

 How many edge from a to a - 0

 How many edge from a to b - 1

 How many edge from b to a - 1

 How many edge from b to b - 0

 THE ADJANCY MATRIX :

0 1
1 0
 THE DEGREE OF a -- 1

 THE DEGREE OF b -- 1

 There is no euler circuit exist

 There is no euler circuit exist

 There is a euler path
```

**16. Given a full m-ary tree with i internal vertices, Write a Program to find the number of leaf nodes.**

**->**

```
#include <iostream>

using namespace std;

int main()

{

    int m, i;

    cout << "Enter the degree of tree : ";

    cin >> m;

    cout << "Enter the value of internal vertices : ";

    cin >> i;

    cout << "The number of leaves is : " << (i * (m - 1) + 1) << endl;

    return 0;

}
```

## Output Of 16th Program in IDE :-



```
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> cd
 ($?) { g++ Q16_mRay.cpp -o Q16_mRay } ; if ($?) { .\Q16_mRay }
Enter the degree of tree : 4
Enter the value of internal vertices :  6
The number of leaves is :  19
PS D:\2. II Semester\2. Discrete Mathematics\1. Practical Of Discrete Mathematics> 
```