Atma Ram Sanatan Dharma College

University of Delhi

Practical File

DISCRETE STRUCTURE

Submitted by :- Priyanshu

College Roll No.:- 18050

BSc.(Hons) Computer Science

Sem-2,year-1

Submitted To :- Mrs.Shalini Gupta ma'am

Department of Computer Science

Q1. 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.

❖ CODE

```cpp
// Ques. 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.
[redacted]

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

int counter = 0;    // making counter a global variable

bool isMember(int a, int A[], int counter){  //function definaton
    for (int i = 0; i < counter; i++)
    {
        if (a == A[i])  // if element found in array then return true
        {
            return true;
        }
        else            // if does not found then return false
            return false;

    }

}
```

```cpp
void powerSet(int A[]){ // function to print all the elements of powerset of A
    unsigned int pow_set_size = pow(2,counter);
    int i,j;

    cout<<"\n P(A):{ ";
    for (int i = 0; i < pow_set_size; i++)
    {
     cout<<"{";
        for (int j = 0; j < pow_set_size; j++)
        {

            if (i & ( 1<< j))
            {
                cout<<A[j];
            }


        }
        cout<<"}, ";
    }
    cout<<" }";

}
```

```cpp
//------ Main ------//
int main(){
    int A[20],i = 0,a;
    char B[20];
    bool flag = true;
    cout<<"(Enter -0  to end input prompt)   \nEnter elements of Set A:";
    while (flag) //----- accepting elements ------//
    {
        cin>>A[i];
        if (A[i] != -0)
        {
            i++;
            counter++;
        }
        else
            flag = false;

    }
    /*/-------  Printing Cardinality of the set -----/ */
    cout<<"\nCardinality of set A is "<<counter<<endl;

    //----- accepting element to be searched in set A -----//
    cout<<"\nEnter the element to be searched among elements of Set A:";
    cin>>a;
    //----- ismember() fxn call. ------//
    //----- and checking weather given element is present in the set or not. -----//
    if(isMember(a,A,counter)){
        cout<<"\nEntered element is a member of set A.";
    }
    else{
        cout<<"\nEntered element is not the member of set A.";
    }

    /*/----- calling powerSet() and
    printing all power sets of set A ------- /*/
    powerSet(A);

    return 0;
}
```

❖ OUTPUT

```
(Enter -0  to end input prompt)
Enter elements of Set A:2
3
4
5
-0

Cardinality of set A is 4

Enter the element to be searched among elements of Set A:8

Entered element is not the member of set A.
 P(A):{ {}, {2}, {3}, {23}, {4}, {24}, {34}, {234}, {5}, {25}, {35}, {235}, {45}, {245}, {345}, {2345},  }
```

```
(Enter -0  to end input prompt)
Enter elements of Set A:2
3
4
5
-0

Cardinality of set A is 4

Enter the element to be searched among elements of Set A:4

Entered element is a member of set A.
 P(A):{ {}, {2}, {3}, {23}, {4}, {24}, {34}, {234}, {5}, {25}, {35}, {235}, {45}, {245}, {345}, {2345},  }
```

Q2. 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.

❖ CODE

```
/*
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.*/
// Author: Priyanshu

#include <iostream>
using namespace std;
class SET
{
   private:
      int i,j;
   public:
   void Subset(int *arrA, int sizeA, int *arrB, int sizeB){
      int c=0;
      for(i=0; i<sizeA; i++)
         for(j=0; j<sizeB; j++)
            if(arrA[i] == arrB[j])
               c++;

      if(c != sizeA)
         cout << "SET A is not a subset of SET B" << endl;
      else
         cout << "SET A is a subset of SET B" << endl;
      int c1=0;
      for(i=0; i<sizeB; i++)
         for(j=0; j<sizeA; j++)
```

```
            if(arrB[i] == arrA[j])
                c1++;
        if(c != sizeB)
            cout << "SET B is not a subset of SET A" << endl;
        else
            cout << "SET B is a subset of SET A" << endl;
        cout << "-----------------------------------------------" << endl;
    }
    void UnionInter(int *setA, int sizeA, int *setB, int sizeB){
        int uSize=sizeA+sizeB;
        int uSet[uSize];
        int unionSet[uSize];
        int iSet[uSize];
        int x=0,y=0;
        for(i=0; i<sizeA; i++){
            uSet[x]=setA[i];
            x++;
        }
        for(i=0; i<sizeB; i++){
            uSet[x]=setB[i];
            x++;
        }
        for(i=0; i<x; i++){
            for(j=i+1; j<x; j++){
                if(uSet[i] == uSet[j]){
                    iSet[y]=uSet[i];
                    y++;
                    for(int k=j; k<x-1; k++)
                        uSet[k]=uSet[k+1];
                        x--;
                }
                else
                    continue;
            }
        }
        cout << "Union of two sets is : {";
        for(i=0; i<x; i++)
            cout << uSet[i] << " ";
        cout << "}";
        cout << endl;
        if(y != 0)
        {
            cout << "Intersection of two sets is : {";
            for(i=0; i<y; i++)
                cout << iSet[i] << " ";
            cout << "}";
        }
        else
            cout << "No intersection found";
        cout << endl;
        cout << "-----------------------------------------------" << endl;
    }
    void Complement(int *setA, int sizeA, int *setB, int sizeB)
    {
```

```
    int sizeU;
    cout << "Enter the no. of elements of universal set : ";
    cin >> sizeU;
    cout << "Enter the elemnts of universal set : ";
    int U[sizeU];
    for(i=0; i<sizeU; i++)
        cin >> U[i];
    int AC[sizeU],p=0,c=0;
    for(i=0; i<sizeU; i++){
        for(j=0; j<sizeA; j++){
            if(U[i] == setA[j])
                c++;
            else
                continue;
        }
        if(c == 0){
            AC[p]=U[i];
            p++;
        }
        c=0;
    }
    cout << endl;
    cout << "Complement of SET A is : {";
    for(i=0; i<p; i++)
        cout << AC[i] << " ";
    cout << "}" << endl;
    int BC[sizeU],q=0,ctr=0;
    for(i=0; i<sizeU; i++){
        for(j=0; j<sizeB; j++){
            if(U[i] == setB[j])
                ctr++;
            else
                continue;
        }
        if(ctr == 0){
            BC[q]=U[i];
            q++;
        }
        ctr=0;
    }
    cout << "Complement of SET B is : {";
    for(i=0; i<q; i++)
        cout << BC[i] << " ";
    cout << "}" << endl;
cout << "-----------------------------------------------" << endl;
}
void setNSymDiff(int *setA, int sizeA, int *setB, int sizeB)
{
    int ABDif[100],q=0,ctr=0;
    for(i=0; i<sizeA; i++){
        for(j=0; j<sizeB; j++){
            if(setA[i] == setB[j])
                ctr++;
            else
```

```
            continue;
        }
        if(ctr == 0){
            ABDif[q]=setA[i];
            q++;
        }
        ctr=0;
    }
    cout << "Set difference A-B is : {";
    for(i=0; i<q; i++)
        cout << ABDif[i] << " ";
    cout << "}" << endl;
    int BADif[100],p=0,c=0;
    for(i=0; i<sizeB; i++){
        for(j=0; j<sizeA; j++){
            if(setB[i] == setA[j])
                c++;
            else
                continue;
        }
        if(c == 0){
            BADif[p]=setB[i];
            p++;
        }
        c=0;
    }
    cout << "Set difference B-A is : {";
    for(i=0; i<p; i++)
        cout << BADif[i] << " ";
    cout << "}" << endl;
    int uSize=q+p;
    int symDif[uSize];
    int x=0,y=0;
    for(i=0; i<q; i++){
        symDif[x]=ABDif[i];
        x++;
    }
    for(i=0; i<p; i++){
        symDif[x]=BADif[i];
        x++;
    }
    cout << "Symmetric difference b/w two sets is : {";
    for(i=0; i<x; i++)
        cout << symDif[i] << " ";
    cout << "}";
    cout << endl;
    cout << "-----------------------------------------------" << endl;
}
void cartesianPro(int *setA, int sizeA, int *setB, int sizeB)
{
    int sizeAB,sizeBA,x=0,y=0;
    sizeAB=sizeA*sizeB;
    sizeBA=sizeAB;
    int AB[sizeAB*2],BA[sizeBA*2];
```

```
        for(i=0; i<sizeA; i++){
            for(j=0; j<sizeB; j++){
                AB[x++]=setA[i];
                AB[x++]=setB[j];
            }
        }
        for(i=0; i<sizeB; i++){
            for(j=0; j<sizeA; j++){
                BA[y++]=setB[i];
                BA[y++]=setA[j];
            }
        }
        cout << "A X B = { ";
        for(i=0; i<x; i++){
            if(i%2 == 0)
                cout << "(";
            cout << AB[i] << " ";
            if(i%2 != 0)
                cout << ")";
        }
        cout << " }" << endl;
        cout << "B X A = { ";
        for(i=0; i<y; i++){
            if(i%2 == 0)
                cout << "(";
            cout << BA[i] << " ";
            if(i%2 != 0)
                cout << ")";
        }
        cout << " }" << endl;
        cout << "-------------------------------------------------" << endl;
    }
};

int main()
{
    cout << endl;
    int i,sizeA,sizeB;
    cout << "Enter the no. of elements in SET A : ";
    cin >> sizeA;
    int arrA[sizeA];
    cout << "Enter the elements : ";
    for(i=0; i<sizeA; i++)
        cin >> arrA[i];
    cout << "Enter the no. of elements in SET B : ";
    cin >> sizeB;
    int arrB[sizeB];
    cout << "Enter the elements : ";
    for(i=0; i<sizeB; i++)
        cin >> arrB[i];
    cout << "-------------------------------------------------" << endl;
    SET ob;
    cout << "\tSUBSET\n" << endl;
    ob.Subset(arrA, sizeA, arrB, sizeB);
```

```
    cout << "\tUNION and INTERSECTION\n" << endl;
    ob.UnionInter(arrA, sizeA, arrB, sizeB);
    cout << "\tCOMPLEMENT\n" << endl;
    ob.Complement(arrA, sizeA, arrB, sizeB);
    cout << "\tSET and SYMMETRIC DIFFERENCE\n" << endl;
    ob.setNSymDiff(arrA, sizeA, arrB, sizeB);
    cout << "\tCARTESIAN PRODUCT\n" << endl;
    ob.cartesianPro(arrA, sizeA, arrB, sizeB);
    return 0;
}
```

❖ OUTPUT

```
Enter the no. of elements in SET A : 3
Enter the elements : 1
2
3
Enter the no. of elements in SET B : 3
Enter the elements : 4
5
6
-------------------------------------------------
        SUBSET

SET A is not a subset of SET B
SET B is not a subset of SET A

Enter the no. of elements of universal set : 6
Enter the elemnts of universal set : 1
2
3
4
5
6

Complement of SET A is : {4 5 6 }
Complement of SET B is : {1 2 3 }
-------------------------------------------------
        SET and SYMMETRIC DIFFERENCE

Set difference A-B is : {1 2 3 }
Set difference B-A is : {4 5 6 }
Symmetric difference b/w two sets is : {1 2 3 4 5 6 }
-------------------------------------------------
        CARTESIAN PRODUCT

A X B = { (1 4 )(1 5 )(1 6 )(2 4 )(2 5 )(2 6 )(3 4 )(3 5 )(3 6 ) }
B X A = { (4 1 )(4 2 )(4 3 )(5 1 )(5 2 )(5 3 )(6 1 )(6 2 )(6 3 ) }
-------------------------------------------------
```

Q3. Create a class RELATION, use Matrix notation to represent a relation. Include functions to check if a relation is reflexive, Symmetric, Anti-symmetric and Transitive. Write a Program to use this class.

❖ CODE

```
/*
Create a class RELATION, use Matrix notation to represent a relation. Include functions to
check if a relation is reflexive, Symmetric, Anti-symmetric and Transitive. Write a Program
to use this class.
*/
//----- Author: Priyanshu -----//


#include<iostream>
#include<stdio.h>
#include<conio.h>
using namespace std;
class RELATION
{
   private:
   int i,j,k,x,y,z,ctr,iA,iB,nA,nR,*A,*R,**RM,**T;
   public:
   void empty();
   int inputSet();
   void inputRelation();
   void printSet();
   void printRelation();
   void Matrix();
   int reflexive();
   int symmetric();
   bool antiSymmetric();
   bool transitive();
};
void RELATION::empty()
{
   cout << "Set A is empty\n";
   printSet();
   cout << "Set A has no member.";
   cout << "\nHence, relation R is empty.\n";
   nR = 0;
   printRelation();
   cout << "Therefore, no matrix notation.";
   cout << "\nRelation R is NOT REFLEXIVE.";
   symmetric();
   antiSymmetric();
   transitive();
}
int RELATION::inputSet()
{
   cout << "Enter the size of SET A : ";
   cin >> nA;
   A = new int[nA];
   if(nA == 0)
      return 1;
   cout << "Enter the elements : ";
   for(i=0; i<nA; i++)
      cin >> A[i];
```

```
}
void RELATION::inputRelation(){
    cout << "Enter the no of relations (R on A) : ";
    cin >> nR;
    R = new int[nR * 2];
    cout << "Enter the relations in pair :\n";
    for(i=0; i<nR*2; i++)
        cin >> R[i];
}
void RELATION::printSet(){
    cout << "A = {";
    for(i=0; i<nA; i++)
        cout << A[i] << " ";
    cout << "}\n";
}
void RELATION::printRelation(){
    cout << "R = {";
    for(i=0; i<nR*2; i++){
        if(i%2 == 0)
            cout << "(";
        cout << R[i] << " ";
        if(i%2 != 0)
            cout << ")";
    }
    cout << "}\n";
}
void RELATION::Matrix(){
    cout << "\nMATRIX NOTATION\n\n";
    RM = new int *[nA];
    for(i=0; i<nA; i++)
        RM[i]=new int[nA];
    for(i=0; i<nA; i++){
        for(j=0; j<nA; j++){
            RM[i][j]=0;
        }
    }
    for(i=0; i<nR*2; i+=2){
        for(j=0; j<nA; j++){
            if(R[i] == A[j]){
                iA=j;
                break;
            }
        }
        for(k=0; k<nA; k++)
        {
            if(R[i+1] == A[k]){
                iB=k;
                break;
            }
        }
        RM[iA][iB]=1;
    }
    cout << "    ";
    for(int x=0; x<nA; x++)
```

```
      cout << " " << A[x] << " ";
   cout << endl << endl;
   for(i=0; i<nA; i++){
      cout << A[i] << "  | ";
      for(j=0; j<nA; j++){
         cout << RM[i][j] << "  ";
      }
      cout << "|";
      cout << endl;
   }
}
int RELATION::reflexive(){
   x=0;
   for(i=0; i<nA; i++){
      if(RM[i][i] == 1)
         x++;
   }
   if(x == nA){
      cout << "\nRelation R is REFLEXIVE.";
      return x = 0;
   }
   else
   {
      cout << "\nRelation R is NOT REFLEXIVE.";
      return x = 1;
   }
}
int RELATION::symmetric()
{
   ctr = 0;
   for(i=0; i<nA; i++){
      for(j=0; j<nA; j++){
         if(RM[i][j] == RM[j][i])
            continue;
         else{
            ctr++;
            break;
         }
      }
   }
   if(ctr != 0)
      cout << "\nRelation R is NOT SYMMETRIC.";
   else
      cout << "\nRelation R is SYMMETRIC.";
   return ctr;
}
bool RELATION::antiSymmetric(){
   bool flag = true;
   for(i=0; i<nR*2; i+=2){
      for(j=0; j<nR*2; j+=2){
         if((R[i] == R[j+1]) && (R[i+1] == R[j]))
            if(R[i] == R[i+1])
            {
               continue;
```

```
            }
            else
            {
               flag = false;
            }
      }
   }
   if(flag != true)
      cout << "\nRelation R is NOT ANTI-SYMMETRIC.";
   else
      cout << "\nRelation R is ANTI-SYMMETRIC.";
   return flag;
}
bool RELATION::transitive(){
   bool flag = true;
   for(i=0; i<nR*2; i+=2){
      for(j=0; j<nR*2; j+=2){
         if(R[i+1] == R[j])
            for(k=0; k<nR*2; k+=2){
               if((R[k] == R[i]) && (R[k+1] == R[j+1])){
                  flag = true;
                  break;
               }
               else
                  flag = false;
            }
      }
   }
   if(flag != true)
      cout << "\nRelation R is NOT TRANSITIVE.";
   else
      cout << "\nRelation R is TRANSITIVE.";
   return flag;
}

int main()
{
   int p = 0;
   RELATION ob;
   p = ob.inputSet();
   if(p == 1)
      ob.empty();
   else
   {
      ob.printSet();
      ob.inputRelation();
      ob.printRelation();
      ob.Matrix();
      ob.reflexive();
      ob.symmetric();
      ob.antiSymmetric();
      ob.transitive();
   }
   return 0;
```

```
}
```

❖ OUTPUT

```
Enter the size of SET A : 4
Enter the elements : 1
2
3
4
A = {1 2 3 4 }
Enter the no of relations (R on A) : 2
Enter the relations in pair :
1
2
1
3
R = {(1 2 )(1 3 )}

MATRIX NOTATION

     1  2  3  4

1  | 0  1  1  0  |
2  | 0  0  0  0  |
3  | 0  0  0  0  |
4  | 0  0  0  0  |

Relation R is NOT REFLEXIVE.
Relation R is NOT SYMMETRIC.
Relation R is ANTI-SYMMETRIC.
Relation R is TRANSITIVE.
```

Q4. Use the functions defined in Ques 3 to find check whether the given relation is:

a) Equivalent, or

b) Partial Order relation, or

c) None

❖ CODE

```
#include<iostream>
#include "RELATIONS.hpp"

using namespace std;

class checkRELATION : public RELATION
{
   public:
   int equivalent(int, int, bool);
   int partialOrder(int, bool, bool);
   void neither(int, int);
};

int checkRELATION::equivalent(int r, int s, bool t)
{
```

```
    if((r == 0) && (s == 0) && (t == true))
        cout << "\nRelation R is EQUIVALENT relation";

    else
        return 0;

    return 1;
}

int checkRELATION::partialOrder(int r, bool a, bool t)
{
    if((r == 0) && (a == true) && (t == true))
        cout << "\nRelation R is PARTIAL ORDER relation";

    else
        return 0;

    return 1;
}

void checkRELATION::neither(int e, int po)
{
    if((e != 1) && (po != 1))
        cout << "\nRelation R is NEITHER equivalent NOR partial order relation";
}
int main()
{
    int p=0,r,s,e,po;
    bool a,t;

    checkRELATION ob1;

    p = ob1.inputSet();

    if(p == 1)
    {
        ob1.empty();
    }

    else
    {
        ob1.printSet();
        ob1.inputRelation();
        ob1.printRelation();
        ob1.Matrix();
        r = ob1.reflexive();
        s = ob1.symmetric();
        a = ob1.antiSymmetric();
        t = ob1.transitive();
    }

    e = ob1.equivalent(r, s, t);
    po = ob1.partialOrder(r, a, t);
    ob1.neither(e, po);
```

```
    return 0;
}
```

❖ OUTPUT

```
Enter the size of SET A : 5
Enter the elements : 1 2 3 4 5
A = {1 2 3 4 5 }
Enter the no of relations (R on A) : 13
Enter the relations in pair :
1 1
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
5 5
R = {(1 1 )(1 3 )(1 4 )(1 5 )(2 2 )(2 3 )(2 4 )(2 5 )(3 3 )(3 4 )(3 5 )(4 4 )(5 5 )}

MATRIX NOTATION

     1  2  3  4  5

1  | 1  0  1  1  1  |
2  | 0  1  1  1  1  |
3  | 0  0  1  1  1  |
4  | 0  0  0  1  0  |
5  | 0  0  0  0  1  |

Relation R is REFLEXIVE.
Relation R is NOT SYMMETRIC.
Relation R is ANTI-SYMMETRIC.
Relation R is TRANSITIVE.
Relation R is PARTIAL ORDER relation
```

```
Enter the size of SET A : 3
Enter the elements : 0 1 2
A = {0 1 2 }
Enter the no of relations (R on A) : 3
Enter the relations in pair :
0 0
1 1
2 2
R = {(0 0 )(1 1 )(2 2 )}

MATRIX NOTATION

     0  1  2

0  | 1  0  0  |
1  | 0  1  0  |
2  | 0  0  1  |

Relation R is REFLEXIVE.
Relation R is SYMMETRIC.
Relation R is ANTI-SYMMETRIC.
Relation R is TRANSITIVE.
Relation R is EQUIVALENT relation
Relation R is PARTIAL ORDER relation
```

Q5. Write a Program to generate the Fibonacci Series using recursion.

❖ CODE

```cpp
//****** Fibonacci_Series ******//
/******* Q5. Write a Program to generate the Fibonacci Series using recursion *****/

#include<iostream>
using namespace std;

/* fibonacci function to print fibonacci series */
int fibonacci(int n){
   if(n == 0 || n == 1)
      return n;
   else
      //----- calling fibonacci function using recursion -----//
      return (fibonacci(n-1)+fibonacci(n-2));
}

//----- Driver code ------//
int main(){
   int n, i = 0;

   cout<<"****** This program prints fibonacci series upto entered n number of terms.******";
   cout<<"\n\nEnter number of terms: ";
   cin>>n;
   while(i<n){
      cout<<fibonacci(i)<<" ";
      i++;
   }
   return 0;
}
```

❖ OUTPUT

```
****** This program prints fibonacci series upto entered n number of terms.******

Enter number of terms: 15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Q6. Write a Program to implement Tower of Hanoi using recursion.

❖ CODE

```cpp
❖  /* **********
❖      Ques. 6 .Write a Program to implement Tower of Hanoi using recursion.
❖      **** Author:Priyanshu *****
❖  ***/
❖  //soln.
❖
❖  // **** TowerOfHanoi using recursion**** //
❖  #include<iostream>
❖  using namespace std;
❖  void towerOfHanoi(int n, char from, char to, char aux){
❖     if (n == 0)
❖     {
❖        return ;
❖     }
```

```
    towerOfHanoi(n-1,from, aux, to);
    cout<<"Move disk "<<n<<" from rod "<<from<<" to rod "<<to<<endl;
    towerOfHanoi(n-1,aux,to,from);

}
//----- Driver code -----//
int main(){
    int n;  // ----- no. of disks ------//
    cout<<"Enter no. of disks: ";
    cin>>n;
    cout<<endl;
    //----- calling towerOfHanoi function -----//
    towerOfHanoi(n,'A','C','B');
    return 0;
}
```

❖ OUTPUT

```
Enter no. of disks: 3

Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
```

Q7. Write a Program to implement binary search using recursion.

```
/****
Quest. 7.
   Write a Program to implement binary search using recursion.
**** */
// Author : Priyanshu
// Soln.

/***** binarySearch *******/

#include<iostream>
using namespace std;

//-------- sorting the entered array first ---------//
void sort(int arr[], int n){
   int counter = 1;
   while (counter < n)
   {
      for (int i = 0; i <  n-counter; i++)
      {
         if (arr[i]>arr[i+1])
         {
```

```
            swap(arr[i],arr[i+1]);
         }

      }
      counter++;

   }
   //------- printing sorted array -------//
   cout<<"\nSorted array: ";
   for (int i = 0; i < n; i++)
   {
      cout<<arr[i]<<" ";
   }
}

//-------- function binarySearch for searching the entered element --------//
int binarySearch(int arr[], int key, int s ,int e){
   int mid;
   if (s>e) //------ if s exceeds e then element isn't present ------//
   {
      cout<<"Element do not exist.";
      return 0;
   }
   else{
      mid = (s + e)/2;

      /*/------ if the mid element is the element
      printing index of mid element -------/*/
      if(arr[mid] == key){
         cout << "Element found at index \n" << mid;
         return 0;
      }

      /*/---------------------- if the element is greater than mid element
       then shifting s to mid + 1 --------------------------/*/
      else if (key > arr[mid]) {
         binarySearch (arr, key, mid+1, e);
      }

      /*/----------- if the element is smaller than mid element
       then shifting e to mid - 1 ----------------/*/
      else if (key < arr[mid]) {
         binarySearch (arr, key, s, mid-1);
      }

   }

}

//------- Driver code ---------//
int main(){
   int n,key;
   cout<<"Enter the size of array: ";
   cin>>n;
```

```
    int s = 0; //------ starting from index 0 ------//
    int e = n-1; //------ ending at last element -------//
    int arr[n];

    //------ accepting array -------//
    cout<<"\nEnter the elements: ";
    for (int i = 0; i < n; i++)
    {
        cin>>arr[i];
    }

    //------- calling sort function ------//
    sort(arr,n);

    //------- accepting element to be searched from user. --------//
    cout<<"\nEnter the element to be searched: ";
    cin>>key;

    //------ calling binarySearch function -------//
    binarySearch(arr, key, s, e);
    return 0;
}
```

❖ OUTPUT

```
Enter the size of array: 5

Enter the elements: 43
5
34
23
67

Sorted array: 5 23 34 43 67
Enter the element to be searched: 43
Element found at index
3
```

Q8. Write a Program to implement Bubble Sort. Find the number of comparisons during each pass and display the intermediate result. Use the observed values to plot a graph to analyse the complexity of algorithm.

❖ CODE:

```
/* **** Ques 8.
Write a Program to implement Bubble Sort. Find the number of comparisons during each
pass and display the intermediate result. Use the observed values to plot a graph to analyse
the complexity of algorithm.
*******
******
Author: Priyanshu
*/

// solution.
```

```cpp
//****** Bubble_sorting ******//

#include<iostream>
using namespace std;

//-------- display function to printing array --------//
void display(int arr[], int n){
   for (int i = 0; i < n; i++)
   {
      cout<<arr[i]<<" ";
   }
}

//--------- bubbleSort function for sorting array --------//
void bubbleSort(int arr[], int n){
   for (int i = 0; i < n-1; i++){
      int counter = 0;
      for (int j = 0; j <n ; j++){
         /*/-------------- if element at jth index is greater than (j+1)th element
          swap the elements at  those index -------------*/
         if (arr[j]>arr[j+1]){
            swap(arr[j],arr[j+1]);
            counter++;
         }

      }
      //-------- printing no. of comparision for each iteration ---------//
      cout<<endl<<"no. of comparison for iteration "<<i+1<<": "<<counter;
   }
}
//------- Driver code --------//
int main(){
   int n; //------ size of array ------//
   cout<<"\nEnter the number of elements in the array:  ";
   cin>>n;
   int arr[n];
   //------ accepting elements in array ------//
   cout<<"\nEnter elements: ";
   for (int i = 0; i < n; i++){
      cin>>arr[i];
   }
   //------- displaying the array entered by user -------//
   cout<<"\nArray entered: ";
   display(arr,n);

   //------ calling bubbleSort function --------//
   bubbleSort(arr,n);

   //------- printing Sorted array -------//
   cout<<endl<<"Sorted array: ";
   display(arr,n);

   return 0;
```

```
}
```

❖ OUTPUT:

```
Enter the number of elements in the array:  5

Enter elements: 50
40
30
20
10

Array entered: 50 40 30 20 10
no. of comparison for iteration 1: 4
no. of comparison for iteration 2: 3
no. of comparison for iteration 3: 2
no. of comparison for iteration 4: 1
Sorted array: 10 20 30 40 50
```

Q9. Write a Program to implement Insertion Sort. Find the number of comparisons during each pass and display the intermediate result. Use the observed values to plot a graph to analyse the complexity of algorithm.

❖ CODE:

```
/*
Ques 9.
Write a Program to implement Insertion Sort. Find the number of comparisons during each
pass and display the intermediate result. Use the observed values to plot a graph to analyse
the complexity of algorithm.

Author: Priyanshu
*/

// Soln. //

//***** Insertion_Sort *****//

#include<iostream>
using namespace std;
//-------- display function to print array -------//
void display(int arr[], int count){
   for (int i = 0; i < count; i++)
   {
      cout<<arr[i]<<" ";
   }

}

//-------- insertionSort function for sorting of array ---------//
 void insertionSort(int arr[] , int n){
   for (int i = 1; i < n; i++)
   {
      int counter = 0;
```

```
        int current = arr[i];
        int j = i-1;
        while (j>=0 && arr[j]>current)
        {
            arr[j+1] = arr[j];
            j--;
            counter++;
        }
        arr[j+1] = current;
        cout<<endl<<"no. of comparison for iteration "<<i<<": "<<counter;
    }

}
//-------- Driver code --------//
int main()
{
    //-------- size of the array ---------//
    int n;
    cout<<"Enter no. of elements: ";
    cin>>n;
    //-------- creating array of size n ---------//
    int arr[n];

    //--------- accepting elements of the array from the user ---------//
    cout<<"\nEnter elements:";
    for (int i = 0; i < n; i++)
    {
        cin>>arr[i];
    }
    //-------- displaying entered array ---------//
    cout<<"\nArray entered: ";
    display(arr, n);

    //------- calling insertionSort function --------//
    insertionSort(arr, n);

    //------- printing sorted array --------//
    cout<<"\nSorted array: ";
    display(arr, n);

    return 0;

}
```

❖ OUTPUT:

```
Enter no. of elements: 5

Enter elements:50
40
30
20
10

Array entered: 50 40 30 20 10
no. of comparison for iteration 1: 1
no. of comparison for iteration 2: 2
no. of comparison for iteration 3: 3
no. of comparison for iteration 4: 4
Sorted array: 10 20 30 40 50
```

Q10. Write a Program that generates all the permutations of a given set of digits, with or without repetition.

❖ CODE:

```
#include<iostream>
#include<stdio.h>
#include<conio.h>
#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 &);

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;
}

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;
    a = t;
}
```

❖ OUTPUT:

```
Enter the size of set: 3
Enter the elements: 4
2
5

Is repetition allowed (Y/N): N
{4 2 5 }{4 5 2 }{2 4 5 }{2 5 4 }{5 2 4 }{5 4 2 }
```

```
Enter the size of set: 3
Enter the elements: 2
4
6

Is repetition allowed (Y/N): Y
{2 2 2 }{2 2 4 }{2 2 6 }{2 4 2 }{2 4 4 }{2 4 6 }{2 6 2 }{2 6 4 }{2 6 6 }{4 2 2 }{4 2 4 }{4 2 6 }{4 4 2 }{4 4 4 }{4 4 6 }{4 6 2
}{4 6 4 }{4 6 6 }{6 2 2 }{6 2 4 }{6 2 6 }{6 4 2 }{6 4 4 }{6 4 6 }{6 6 2 }{6 6 4 }{6 6 6 }
```

Q11. Write a Program to calculate Permutation and Combination for an input value n and r using recursive formula of nCr and nPr.

❖ CODE:

```
/*
// -----Q_11. -----//
Write a Program to calculate Permutation and Combination for an input value n and r using
recursive formula of nCr and nPr .
***Author: Priyanshu
****/

#include<iostream>
using namespace std;

//------ permutation function to calculate nPr. -------//
int permutation(int n, int r){
    int res = 0;
    if (r == 0) {
        return 1;
```

```
    }
    else {
        //----- recursive way to call the fxn again ------//
        res = permutation(n, r - 1)  * (n - r + 1);
    }
    //----- returning the calculated result as res -----//
    return res;
}

//------- combination function to calculate nCr ------//
int combination(int n, int r){
    int res = 0;
    if (r == 0) { // ----- if  r = 0, return 1 as result (res) -----//
        return 1;
    }
    else {
        //----- recursive way to call the fxn again ------//
        res = combination(n, r - 1) * (n - r + 1) / r;
    }
    //----- returning the calculated result as res -----//
    return res;
}

//------ Driver code -------//
int main(){

    int n,r;
    // --- accepting values of n and r ------//
    cout<<"\nEnter n:";
    cin>>n;
    cout<<"Enter r:";
    cin>>r;

    //----- printing the result of nPr -----//
    cout<<n<<"P"<<r<<" = "<<permutation(n,r)<<endl;

    //----- printing the result of nCr -----//
    cout<<n<<"C"<<r<<" = "<<combination(n,r);

    return 0;
}
```

❖ OUTPUT:

```
Enter n:5
Enter r:3
5P3 = 60
5C3 = 10
```

Q12. 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.

❖ CODE:

```
/* //----- Question 12 -----//
*** 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.
*/
// ----- Author: Priyanshu ----- //

#include<iostream>
using namespace std;
//--------- function bForce delcalaration -------//
void bForce(int*, int, int*, int, int, int, int&);
//------- Driver code -------//
int main(){
   int n, C, counter = 0, size = 11;
   int arr[size], data[100] = {0};
   cout << "\nFinding solutions to x1 + x2 + ... + xn = C\n";
   cout << "Enter the value of n: ";
   cin >> n;
   for (int i=0; i <= 10; i++)
      arr[i] = i;
   //------- Accepting C -------//
   cout << "Enter the sum constant (C <= 10): ";
   cin >> C;
   //-------- Printing solutions -------//
   cout << "Possible solutions [ ";
   for(int i=0; i<n; i++)
      cout << "x" << i+1 << " ";
   cout << " ] :" << endl;
   bForce(arr, size, data, n-1, 0, C, counter);
   cout << "\nThere are "<< counter << " Solutions\n";
   return 0;
}
//---------- function bForce defination ------------//
void bForce(int* arr, int size, int* data, int last, int index, int C, int &counter){
   for(int i=0; i<size; i++){
      data[index] = arr[i];
      if(index == last){
         int sum = 0;
         for(int j=0; j<index+1; j++)
            sum += data[j];

         if(sum == C){
            cout << "[ ";
            for(int j=0; j<index+1; j++)
               cout << data[j] << " ";
            cout << "] ";
            counter++;
         }
```

```
    }

    else
       bForce(arr, size, data, last, index+1, C, counter);
  }
}
```

❖ OUTPUT:

```
Finding solutions to x1 + x2 + ... + xn = C
Enter the value of n: 3
Enter the sum constant (C <= 10): 10
Possible solutions [ x1 x2 x3  ] :
[ 0 0 10 ] [ 0 1 9 ] [ 0 2 8 ] [ 0 3 7 ] [ 0 4 6 ] [ 0 5 5 ] [ 0 6 4 ] [ 0 7 3 ] [ 0 8 2 ] [ 0 9 1 ] [ 0 10 0 ] [ 1 0 9 ] [ 1 1
 8 ] [ 1 2 7 ] [ 1 3 6 ] [ 1 4 5 ] [ 1 5 4 ] [ 1 6 3 ] [ 1 7 2 ] [ 1 8 1 ] [ 1 9 0 ] [ 2 0 8 ] [ 2 1 7 ] [ 2 2 6 ] [ 2 3 5 ] [
2 4 4 ] [ 2 5 3 ] [ 2 6 2 ] [ 2 7 1 ] [ 2 8 0 ] [ 3 0 7 ] [ 3 1 6 ] [ 3 2 5 ] [ 3 3 4 ] [ 3 4 3 ] [ 3 5 2 ] [ 3 6 1 ] [ 3 7 0 ]
 [ 4 0 6 ] [ 4 1 5 ] [ 4 2 4 ] [ 4 3 3 ] [ 4 4 2 ] [ 4 5 1 ] [ 4 6 0 ] [ 5 0 5 ] [ 5 1 4 ] [ 5 2 3 ] [ 5 3 2 ] [ 5 4 1 ] [ 5 5
0 ] [ 6 0 4 ] [ 6 1 3 ] [ 6 2 2 ] [ 6 3 1 ] [ 6 4 0 ] [ 7 0 3 ] [ 7 1 2 ] [ 7 2 1 ] [ 7 3 0 ] [ 8 0 2 ] [ 8 1 1 ] [ 8 2 0 ] [ 9
 0 1 ] [ 9 1 0 ] [ 10 0 0 ]
There are 66 Solutions
```

Q13. 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 f) Exclusive NOR

b) Disjunction g) Negation

c) Exclusive OR h) NAND

d) Conditional i) NOR

e) Bi-conditional

❖ CODE:

```
/*
Ques. 13.. 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 f) Exclusive NOR
b) Disjunction g) Negation
c) Exclusive OR h) NAND
d) Conditional i) NOR
e) Bi - conditonal
**
**Author : Priyanshu
*/


#include<iostream>
using namespace std;

// -------- printing negation of X and Y --------//
void negation(int arrA[], int arrB[]){
   cout<<endl<<"----- Negation -----";
   cout<<endl<<" "<<"\t"<<"_"<<"\t"<<" "<<"\t"<<"_";
   cout<<endl<<"X"<<"\t"<<"X"<<"\t"<<"Y"<<"\t"<<"Y";
   for (int i = 0; i < 4; i++)
   {
      cout<<endl<<arrA[i]<<"\t"<<(!arrA[i])<<"\t"<<arrB[i]<<"\t"<<(!arrB[i]);
   }
```

```
}
// -------- printing cojunction of X and Y --------//
void cojunction(int arrA[], int arrB[]){
   cout<<endl<<endl<<"----- Conjuction ----- ";
   cout<<endl<<"X"<<"\t"<<"Y"<<"\t"<<"F";
   for (int i = 0; i < 4; i++)
   {
      cout<<endl<<arrA[i]<<"\t"<<arrB[i]<<"\t"<<(arrA[i] or arrB[i]);
   }

}
// -------- printing disjunction of X and Y --------//
void disjunction(int arrA[], int arrB[]){
   cout<<endl<<endl<<"----- Disjunction ----- ";
   cout<<endl<<"X"<<"\t"<<"Y"<<"\t"<<"F";
   for (int i = 0; i < 4; i++)
   {
      cout<<endl<<arrA[i]<<"\t"<<arrB[i]<<"\t"<<(arrA[i] and arrB[i]);
   }

}
// -------- printing conditional of X and Y --------//
void Conditional(int arrA[], int arrB[]){
   int res = 0;
   cout<<endl<<endl<<"----- Conditional ----- ";
   cout<<endl<<"X"<<"\t"<<"Y"<<"\t"<<"F = X => Y";
   for (int i = 0; i < 4; i++)
   {
      if (arrA[i]==arrB[i])
      {
         res = 1;
      }
      else if (arrA[i] == 0 && arrB[i] == 1)
      {
         res = 1;
      }
      else
         res = 0;

      cout<<endl<<arrA[i]<<"\t"<<arrB[i]<<"\t"<<res;
   }

}
// -------- printing Bi-conditional of X and Y --------//
void bi_conditional(int arrA[], int arrB[]){
   cout<<endl<<endl<<"----- Bi_conditional ----- ";
   cout<<endl<<"X"<<"\t"<<"Y"<<"\t"<<"F = X <-> Y";
   for (int i = 0; i < 4; i++)
   {
      cout<<endl<<arrA[i]<<"\t"<<arrB[i]<<"\t"<<(!(arrA[i] xor arrB[i]));
   }

}
// -------- printing XOR of X and Y --------//
```

```
void Xor(int arrA[], int arrB[]){
   cout<<endl<<endl<<"----- XOR ----- ";
   cout<<endl<<"X"<<"\t"<<"Y"<<"\t"<<"F";
   for (int i = 0; i < 4; i++)
   {
      cout<<endl<<arrA[i]<<"\t"<<arrB[i]<<"\t"<<(arrA[i] xor arrB[i]);
   }

}
// -------- printing XNOR of X and Y --------//
void Xnor(int arrA[], int arrB[]){
   cout<<endl<<endl<<"----- XNOR ----- ";
   cout<<endl<<"X"<<"\t"<<"Y"<<"\t"<<"F";
   for (int i = 0; i < 4; i++)
   {
      cout<<endl<<arrA[i]<<"\t"<<arrB[i]<<"\t"<<(!(arrA[i] xor arrB[i]));
   }
}
// -------- printing NAND of X and Y --------//
void Nand(int arrA[], int arrB[]){
   cout<<endl<<endl<<"----- NAND ----- ";
   cout<<endl<<"X"<<"\t"<<"Y"<<"\t"<<"F";
   for (int i = 0; i < 4; i++)
   {
      cout<<endl<<arrA[i]<<"\t"<<arrB[i]<<"\t"<<(!(arrA[i] and arrB[i]));
   }
}
// -------- printing NOR of X and Y --------//
void Nor(int arrA[], int arrB[]){
   cout<<endl<<endl<<"----- NOR ----- ";
   cout<<endl<<"X"<<"\t"<<"Y"<<"\t"<<"F";
   for (int i = 0; i < 4; i++)
   {
      cout<<endl<<arrA[i]<<"\t"<<arrB[i]<<"\t"<<(!(arrA[i] or arrB[i]));
   }
}

int main(){
   int arrX[4], arrY[4];
   //------- accepting truth values of X in array arrX[] -------//
   cout<<"Enter truth values of X(0 or 1): ";
   for (int i = 0; i < 4; i++)
   {
      cin>>arrX[i];
   }
   //------- accepting truth values of Y in array arrY[] -------//
   cout<<"Enter truth values of Y(0 or 1): ";
   for (int i = 0; i < 4; i++)
   {
      cin>>arrY[i];
   }
   //------ calling negation function ------//
   negation(arrX,arrY);
```

```
  //------ calling cojunction function ------//
  cojunction(arrX,arrY);

  //------ calling disjunction function ------//
  disjunction(arrX,arrY);

  //------ calling conditional function ------//
  Conditional(arrX,arrY);

  //------ calling Bi-conditional function ------//
  bi_conditional(arrX,arrY);

  //------ calling Xor function ------//
  Xor(arrX,arrY);

  //------ calling Xnor function ------//
  Xnor(arrX,arrY);

  //------ calling Nand function ------//
  Nand(arrX,arrY);

  //------ calling Nor function ------//
  Nor(arrX,arrY);

  return 0;

}
```

❖ OUTPUT:

```
Enter truth values of X(0 or 1): 0
0
1
1
Enter truth values of Y(0 or 1): 0
1
0
1

----- Negation -----
X       X       Y       Y
0       1       0       1
0       1       1       0
1       0       0       1
1       0       1       0

----- Conjuction -----
X       Y       F
0       0       0
0       1       1
1       0       1
1       1       1
```

```
----- Disjunction -----
X       Y       F
0       0       0
0       1       0
1       0       0
1       1       1

----- Conditional -----
X       Y       F = X => Y
0       0       1
0       1       1
1       0       0
1       1       1

----- Bi_conditional -----
X       Y       F = X <-> Y
0       0       1
0       1       0
1       1       1

----- XOR -----
X       Y       F
0       0       0
0       1       1
1       0       1
1       1       0

----- XNOR -----
X       Y       F
0       0       1
0       1       0
1       0       0
1       1       1
```

```
----- NAND -----
X       Y       F
0       0       1
0       1       1
1       0       1
1       1       0

----- NOR -----
X       Y       F
0       0       1
0       1       0
1       0       0
1       1       0
```

Q15. Write a Program to store a function (polynomial/exponential), and then evaluate the polynomial.

❖ CODE:

```
/* //----- Question 15 -------//
. 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)).
**
**Author: Priyanshu
*/


#include<iostream>
using namespace std;

//------ function fx() to calculate the value of f(n) ------//
int fx(int Arr[],int x){
   int sum = 0;
   for(int i = 0; i < 3; i++)
   {
      //----- First it will calculate ((coeff. of (n*n*n)) * n*n*n) and store it in sum ------//
      if (i == 0)
      {
         sum += Arr[0] * (x * x * x);
      }

      //----- Then it will calculate ((coeff. of n) * n ) and add it into sum ------//
      else if(i == 1)
      {
         sum += Arr[1] * x;
      }
      //----- Then the constant will be added into sum ------//
      else
         sum += Arr[2];
   }
   return sum;
}
//------- Driver code -------//
int main(){

   int A[3];
   // ----- accepting coefficient of n*n*n (n cube), n and constant C ------//
   cout<<"                    3"<<endl;
   cout<<"Enter coefficient of n   , n and constant C:";
   for (int i = 0; i < 3; i++)
   {
      cin>>A[i];
   }

   //------- printing the entered values as f(n) ------//
   cout<<"        3  "<<endl;
   cout<<"f(n) = "<<A[0]<<"n "<<"+ "<<A[1]<<"n "<<"+ "<<A[2];
```

```
    //----- Asking user for value of n ------//
    int n;
    cout<<endl<<"Enter n:";
    cin>>n;

    //----- printing f(n) and value of f(n) ------//
    cout<<"f("<<n<<") = "<<fx(A,n);
    return 0;
}
```

❖ OUTPUT:

```
                        3
Enter coefficient of n    , n and constant C:4

2
9
          3
f(n) = 4n + 2n + 9
Enter n:5
f(5) = 519
```

Q16. Write a Program to represent Graphs using the Adjacency Matrices and check if it is a complete graph.

❖ CODE:

```
/*
||********  Q_16  *********||
Write a Program to represent Graphs using the Adjacency Matrices and check if it is a
complete graph.
*/

#include<iostream>
using namespace std;

//----- Driver code ------//
int main(){
    //----- accepting number of vertices and edges ------//
    int v, e;
    cout<<"Enter the no. of vertices: ";
    cin>>v;
    cout<<"\nEnter the no. of edges: ";
    cin>>e;

    //----- creating 2d array to store graph ------//
    //------ initializing all elemets to 0 -------//
    int adj[v][v];
    for(int i = 0; i < v; i++)
    {
        for (int j = 0; j < v; j++)
        {
            adj[i][j] = 0;
        }
```

```
    }

    //----- asking user to enter the adjacent vertices and storing 1 at that index ------//
    for(int i = 0; i < 2*e; i++)
    {
        int a ,b;
        cout<<"\nEnter adjacent vertices: ";
        cin>>a;
        cout<<" is adjacent to ";
        cin>>b;
        adj[a-1][b-1] = 1;
    }

    //------- Printing the adjacency matrix ------//
    cout<<"\n\aAdjacency Matrix:\n";
    for(int i = 0; i < v; i++)
    {
        for (int j = 0; j < v; j++)
        {
            cout<<adj[i][j]<<" ";
        }
        cout<<endl;
    }
    //------ Checking if the entered graph is complete or not -----//
    if(e == (v*(v-1)/2))
    {
        cout<<"\n It is a complete graph.";
    }
    else{
        cout<<"\n It is not a complete graph.";
    }
    return 0;

}
```

❖ OUTPUT:

```
Enter the no. of edges: 6

Enter adjacent vertices: 1
 is adjacent to 2

Enter adjacent vertices: 1
 is adjacent to 3

Enter adjacent vertices: 2
 is adjacent to 1

Enter adjacent vertices: 2
 is adjacent to 4

Enter adjacent vertices: 3
 is adjacent to 1

Enter adjacent vertices: 3
 is adjacent to 4

Enter adjacent vertices: 4
 is adjacent to 2

Enter adjacent vertices: 4
 is adjacent to 3

Enter adjacent vertices: 4
 is adjacent to 3

Enter adjacent vertices: 3
 is adjacent to 5

Enter adjacent vertices: 5
 is adjacent to 3
Enter adjacent vertices: 5
 is adjacent to 4

Adjacency Matrix:
0 1 1 0 0
1 0 0 1 0
1 0 0 1 1
0 1 1 0 0
0 0 1 1 0

 It is not a complete graph.
```

Q.17. Write a Program to accept a directed graph G and compute the in-degree and out-degree of each vertex.

❖ CODE:

```
/*
/* -------- Question 17 -------//
Write a Program to accept a directed graph G and compute the in-degree and out-degree of
each vertex.
*****
*** Priyanshu
**
*/

#include<iostream>
using namespace std;
//------ Driver code ------//
int main(){

   //------- accepting number of vertices and edges -------//
   int v, e, in_d, out_d;
   cout<<"Enter no. of vertices: ";
   cin>>v;
```

```
    cout<<"\nEnter no. of edges: ";
    cin>>e;
    //------- Creating an array to store Adjacency matirx of a directed graph --------//
    int arr[v][v];
    //------- Initializing each element to 0 -------//
    for(int i = 0; i < v; i++)
    {
        for (int j = 0; j < v; j++)
        {
            arr[i][j] = 0;
        }

    }

    //----- Asking user to enter the directions of graph -------//
    for(int i = 0; i < e; i++)
    {
        int a ,b;
        cout<<"\nEnter direction of  vertices: ";
        cin>>a;
        cout<<" is directed to ";
        cin>>b;
        arr[a-1][b-1] = 1;
    }
    //------ code to find In degree of each vertex ------//
    //----- sum of column gives In-degree of the vertex ------//
    for (int i = 0; i < v ; i++)
    {
        in_d = 0;
        cout<<"\nIn-deg("<<i+1<<"): ";
        for (int j = 0; j < v; j++)
        {
            in_d += arr[j][i];
        }
        cout<<in_d;

    }
    cout<<endl;
    //------- code to find Out-degree of each vertex -------//
    //----- sum of row gives Out-degree of the vertex ------//
    for (int i = 0; i < v ; i++)
    {
        out_d = 0;
        cout<<"\nOut-deg("<<i+1<<"): ";
        for (int j = 0; j < v; j++)
        {
            out_d += arr[i][j];
        }
        cout<<out_d;

    }

    return 0;
}
```

❖ OUTPUT:

```
Enter no. of vertices: 5

Enter no. of edges: 6

Enter direction of  vertices: 1
 is directed to 3

Enter direction of  vertices: 2
 is directed to 1

Enter direction of  vertices: 2
 is directed to 4

Enter direction of  vertices: 3
 is directed to 5

Enter direction of  vertices: 4
 is directed to 3

Enter direction of  vertices: 5
 is directed to 4

In-deg(1): 1
In-deg(2): 0
In-deg(3): 2
In-deg(4): 2
In-deg(5): 1

Out-deg(1): 1
Out-deg(2): 2
Out-deg(3): 1
Out-deg(4): 1
Out-deg(5): 1
```

Q.19. 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).

❖ CODE:

```
/* /----- Question no. 19 -------/ */
/****
 * 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).
*****
Author: Priyanshu
******/

#include<iostream>
using namespace std;

int main(){
   //------- Accepting number of veritces -------//
   int v;
   cout<<"Enter the number of vertices:";
   cin>>v;

   int arr[v][v]; //--- matrix declared ----//
```

```
    //------- Entering adjacency matrix ----------//
    cout<<"\nEnter the adjacency matrix(row wise):";
    for (int i = 0; i < v; i++){
        for (int j = 0; j < v; j++){
            cin>>arr[i][j];
        }
        cout<<endl;

    }

    int degree, order = 0;
    for (int i = 0; i < v; i++){
        degree = 0;
        for (int j = 0; j < v; j++){
            degree += arr[i][j];

            if (degree % 2 != 0){
                order++;
            }

        }

    }
    //--------- Checking the order ---------//
    if (order == 0){
        cout<<"\nGraph has an Eularian circuit.";
    }
    else if (order == 2){
        cout<<"\nGraph has an Eularian path.";
    }
    else{
        cout<<"\nGraph is not Eularian.";
    }

    return 0;
}
```

❖ OUTPUT:

```
Enter the number of vertices:5

Enter the adjacency matrix(row wise):0
1
1
0
0

1
0
0
1
0

1
0
0
1
1

0
1
1
0
1

0
0
1
1
0


Graph is not Eularian.
```

Q.20. Given a full m-ary tree with i internal vertices, Write a Program to find the number of leaf nodes.

❖ CODE:

```
/*
*//***** Q_20*****//*
Given a full m-ary tree with i internal vertices, Write a Program to find the number of leaf
nodes.
****
*** Author: Priyanshu
*/

#include<iostream>
using namespace  std;

//------  calLeaf function defination ------//
//------ formula: when m and i is given, l = (((m-1)*i)+1) -----//
int calLeaf(int m, int i){
   int l = ((m - 1)*i) + 1;
   return l;
```

```
}
//------ Driver code -------//
int main(){
   //------ accepting values of m and i from user ------//
   int m,i;
   cout<<"Enter value of m: ";
   cin>>m;
   cout<<"\nEnter number of internal vertices: ";
   cin>>i;
   //------calling calLeaf() function which returns number of leaf nodes -------//
   cout<<"\nThis full "<<m<<"-ary tree have "<<calLeaf(m,i)<<" leaf nodes.";
   return 0;
}
```

❖ OUTPUT: