Program 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.

Ans: #include<iostream>

#include<conio.h>

#include<math.h>

using namespace std;

class set

{

int *ar;

int size;

public:

set()

{

size=10;

ar=new int[size];

}

void input();

void display();

void cardinality();

void unique();

int ismember();

```
    void powerset();
};

void set::input()
{
        cout<<"enter the size of array"<<endl;
        cin>>size;
        ar=new int[size];
        cout<<"enter the array elements"<<endl;
        for(int i=0;i<size;i++)
        cin>>ar[i];
}

void set::display()
{
    cout<<"the given set is {";
    for(int i=0;i<size;i++)
    {
        if(i == 0)
            cout<<ar[i];
        else
            cout<<"," <<ar[i];
    }
    cout<<"}"<<endl;
```

```
}

void set::cardinality()
{
        cout<<"The cardinality of given set is "<<size<<endl;
}

void set::unique()
{
   int i,j,l;
   for(i=0;i<size;++i)
   {
      for(j=i+1;j<size;)
      {
         if(ar[i]==ar[j])
         {
            for(l=j;l<size-1;++l)
               ar[l]=ar[l+1];
               --size;
         }
         else
            ++j;
      }
   }
```

```
}

void set::powerset()
{
   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<<ar[j]<<",";
         temp=temp>>1;
      }
      cout<<"\b}";
   }
        cout<<" }";
}

int set::ismember()
{
```

```
    int e,flag=0;
    cout<<"enter the element to be search"<<endl;
    cin>>e;

    for(int i=0;i<size;i++)
    if(e==ar[i])
    {
       flag=1;
       break;
    }
    return flag;
}

int main()
{
        int ch;
        char ch1;
        set a;
        a.input();
        a.unique();
        a.display();
        a.cardinality();
        do
        {
```

```
        cout<<"Enter your choice"<<endl;
        cout<<"1.Power set"<<endl<<"2.Is
member"<<endl<<"3.Exit"<<endl;
        cin>>ch;
                switch(ch)
                {
                        case 1: a.powerset();
                                break;
                        case 2: if(a.ismember())
                                cout<<"given element belong to set "<<endl;
                            else
                                cout<<"given element not belong to
set"<<endl;
                            break;
                        case 3:exit(0);

                        default:cout<<"wrong choice!!..";
                                        break;
                }
        cout<<endl<<"Do you want to enter more"<<endl;
        cin>>ch1;
    }while((ch1=='y')||(ch1=='Y'));
    return 0;
}
```

Program 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.

Ans:

```
#include<iostream>
#include<conio.h>
#include<math.h>
using namespace std;
class set
{
   int *ar;
   int size;
   public:
   set()
   {
      size=10;
      ar=new int[size];
   }
   void input();
   void display();
```

```
    void setunion(set &a,set &b);
    void unique();
    void intersection(set &a,set &b);
    int ismember(int e);
    int subset(set &b);
    void complement(set &a);
    void cartesian(set &a);
    void symdif(set &a, set &b);
    void diff(set &a, set &b);
};

void set::input()
{
        cin>>size;
        ar=new int[size];
        cout<<"enter the array elements"<<endl;
        for(int i=0;i<size;i++)
        cin>>ar[i];
}

void set::display()
{
    cout<<"the given set is {";
    for(int i=0;i<size;i++)
```

```
    {
        if(i==0)
            cout<<ar[i];
        else
            cout<<","<<ar[i];
    }
    cout<<"}";
    cout<<endl;
}

void set::setunion(set &a,set &b)
{
        int i=0,j=0,k=0;
        while(i<a.size && j<b.size)
        {
       if(a.ar[i]<b.ar[j])
                        ar[k++]=a.ar[i++];
                else
                        ar[k++]=b.ar[j++];
        }

    while(i<a.size)
        {
                ar[k++]=a.ar[i++];
```

```
    }

    while(j<b.size)
    {
                ar[k++]=b.ar[j++];
         }

    size=a.size+b.size;
}

void set::unique()
{
    int i,j,l;
    for(i=0;i<size;++i)
    for(j=i+1;j<size;)
    {
       if(ar[i]==ar[j])
       {
          for(l=j;l<size-1;++l)
             ar[l]=ar[l+1];
             --size;
                }
       else
          ++j;
```

```
    }
}

void set::intersection(set &a,set &b)
{
        size=0;
        for(int i=0;i<a.size;i++)
    {
       for(int j=0;j<b.size;j++)
       {
          if(a.ar[i]==b.ar[j])
          {
             ar[size]=a.ar[i];
             size++;
          }
       }
    }
}

int set::subset(set &b)
{
    int i=0,j=0;
    for(i=0;i<b.size;i++)
    {
```

```
        for(j=0;j<size;j++)
        {
            if(b.ar[i]==ar[i])
            break;
        }
        if(j==size)
            return 0;
    }
    return 1;
}

void set::complement(set &a)
{
    int x[10]={1,2,3,4,5,6,7,8,9,10};
    int p[20],ctr=0;
    for(int i=0;i<a.size;i++)
    {
        for(int j=0;j<10;j++)
        {
            if(x[j]==a.ar[i])
            {
                i++;
                continue;
            }
```

```
            else
            {
                p[ctr]=x[j];
                ctr++;
            }
        }
    }
    cout<<"complement is {";
    for(int i=0;i<ctr;i++)
    {
        if(i==0)
            cout<<p[i];
        else
            cout<<","<<p[i];
    }
}

void set::cartesian(set &b)
{
    cout<<"{";
    for(int i=0;i<size;i++)
    {
        for(int j=0;j<b.size;j++)
            cout<<"("<<ar[i]<<","<<b.ar[j]<<")"<<",";
```

```
    }
    cout<<"}";
}

void set::symdif(set &a, set &b)
{
    int c=0; int p[10];int flag;
    for(int i=0;i<a.size;i++)
    {
        flag=0;
        for(int j=0;j<b.size;j++)
        {
            if(a.ar[i]==b.ar[j])
            {
                flag=0;
                break;
            }
            else
            {
                flag=1;
            }
        }
        if(flag==1)
        {
```

```
        p[c]=a.ar[i];
        c++;
    }
}
for(int k=0;k<b.size;k++)
{
    flag=0;
    for(int h=0;h<a.size;h++)
    {
        if(b.ar[k]==a.ar[h])
        {
            flag=0;
            break;
        }
        else
        {
            flag=1;
        }
    }
    if(flag==1)
    {
        p[c]=b.ar[k];
        c++;
    }
```

```
    }
    cout<<"The symmetric differnce is "<<"{";
    for(int f=0;f<c;f++)
    cout<<p[f]<<",";
    cout<<"}";
}

void set::diff(set &a, set &b)
{
    int c=0; int p[10];int flag;
    for(int i=0;i<a.size;i++)
    {
        flag=0;
        for(int j=0;j<b.size;j++)
        {
            if(a.ar[i]==b.ar[j])
            {
                flag=0;
                break;
            }
            else
            {
                flag=1;
            }
```

```
        }
        if(flag==1)
        {
            p[c]=a.ar[i];
            c++;
        }
    }
    cout<<"The differnce is "<<"{";
    for(int f=0;f<c;f++)
    cout<<p[f]<<",";
    cout<<"}";
}

int main()
{
    set a;
    set b;
    set c;
    set d;
    set e;
    set f;
    set g;
    set h;
```

```
int ch,ch2,ch3;
char ch1;
cout<<"enter the size of Ist array"<<endl;
a.input();
a.unique();
a.display();
cout<<"enter the size of IInd array"<<endl;
b.input();
b.unique();
b.display();
do
{
cout<<endl<<"Enter your choice"<<endl;

cout<<"1.Union"<<endl<<"2.Intersection"<<endl<<"3.subset"<<endl
<<"4.Complement"<<endl<<"5.cartesian
product"<<endl<<"6.Symmetric
differnce"<<endl<<"7.Differnce"<<endl<<"8.exit"<<endl;
cin>>ch;
switch(ch)
{
   case 1: c.setunion(a,b);
           c.unique();
           c.display();
           break;
```

```
    case 2: d.intersection(a,b);
            d.display();
            break;
    case 3: if(b.subset(a))
               cout<<"A is Subset of B"<<endl;
            else
               if(a.subset(b))
                  cout<<"B is subset of A"<<endl;
               else
                  cout<<"A and B are not subset of each other"<<endl;
            break;
    case 4: cout<<"what you want to find complement of 1)A or
2)B"<<endl;
            cin>>ch2;
            if(ch2==1)
               e.complement(a);
            else
               if(ch2==2)
               {
                  e.complement(b);
               }
               else
                  cout<<"wrong choice";
            break;
```

```
    case 5: cout<<"The cartesian product is ";
            a.cartesian(b);
            break;
    case 6: f.symdif(a,b);
            break;
    case 7: cout<<"Enter your choice "<<"1.A-B"<<endl<<"B-
A"<<endl;
            cin>>ch3;
            if(ch3==1)
            {
               g.diff(a,b);
            }
            else if(ch3==2)
            {
               h.diff(b,a);
            }
            else
               cout<<"Wrong choice ";
            break;
    default:cout<<"wrong choice";
            break;
  }

  cout<<"Do you want to enter more"<<endl;
```

```
    cin>>ch1;
    }while((ch1=='y')||(ch1=='Y'));
    return 0;
}
```

Program 3: 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.

Ans:

```
#include<iostream>
using namespace std;
class set
{
    int **ar;
    int size;
    public:
    void setsize();
    void enter();
    int reflexive();
    void display();
    bool symmetric();
    int antisym();
    int transitive();
};

void set::setsize()
```

```
{
   cout<<"Enter the size "<<endl;
   cin>>size;
   ar=new int*[size];
   for(int i=0;i<size;i++)
   {
      ar[i]=new int[size];
   }
   for(int i=0;i<size;i++)
      for(int j=0;j<size;j++)
      {
         ar[i][j]=0;
      }
}

void set::enter()
{
   int a,b,n;
   cout<<"Enter the no of relations"<<endl;
   cin>>n;
   for(int i=0;i<n;i++)
   {
      cout<<"enter the element"<<endl;
      cin>>a>>b;
```

```
        if((a==0)||(b==0)||(a>size)||(b>size))
        {
            cout<<"wrong choice"<<endl;
            exit(0);
        }
        else
        {
            ar[a-1][b-1]=1;
        }
    }
}

int set::reflexive()
{
    int flag=0;
    for(int i=0;i<size;i++)
    {
        if(ar[i][i]==1)
        {
            flag=1;
        }
        else
        {
            flag=0;
```

```
            break;
        }
    }
    return flag;
}

void set::display()
{
    cout<<"The relation in matrix form is "<<endl;
    for(int i=0;i<size;i++)
    {
        for(int j=0;j<size;j++)
        cout<<ar[i][j]<<" ";
        cout<<endl;
    }
}

bool set::symmetric()
{
    int flag;
    for(int i=0;i<size;i++)
    {
        for(int j=i+1;j<size;j++)
        {
```

```
            if((ar[i][j]!=ar[j][i]))
            return false;
            break;
        }
    }
    return true;
}

int set:: antisym()
{
    int flag=0;
    for(int i=0;i<size;i++)
    for(int j=i+1;j<size;j++)
    {
        if(((ar[i][j])&&(ar[j][i]))==1)
        {
            flag=1;
            break;
        }
        else
        {
            flag=0;
        }
    }
```

```
    return flag;
}

int set::transitive()
{
    int flag=1;
    for(int i=0;i<size;i++)
    {
        for(int j=0;j<size;j++)
        {
            if(ar[i][j]==1)
            {
                for(int k=0;k<size;k++)
                {
                    if((ar[j][k]==1)&&(ar[i][k]!=1))
                    flag=0;
                }
            }
        }
    }
    return  flag;
}

int main()
```

```
{
   int ch;
   char ch1;
   set a;
   a.setsize();
   a.enter();
   a.display();

cout<<"1.Reflexive"<<endl<<"2.Symmetric"<<endl<<"3.antisymmetri
c"<<endl<<"4.transitive"<<endl<<"5.exit()"<<endl;
   do
   {
      cout<<"enter your choice ";
      cin>>ch;
      switch(ch)
      {
         case 1: if(a.reflexive())
                    cout<<"the given relation is reflexive"<<endl;
                 else
                    cout<<"The given relation is not reflexive"<<endl;
                 break;
         case 2: if(a.symmetric())
                    cout<<"The relation is symmetric"<<endl;
                 else
```

```
                cout<<"The relation is not symmetric"<<endl;
            break;
        case 3: if(a.antisym())
                cout<<"The given relation is not antisymmetric"<<endl;
            else
                cout<<"The given relation is  antisymmetric"<<endl;
            break;
        case 4: if(a.transitive())
                cout<<"the given relation is transitive"<<endl;
            else
                cout<<"the given relation is not transitive"<<endl;
            break;
        case 5:exit(0);

        default: cout<<"wrong choice"<<endl;
            break;
      }
      cout<<"Do you want to enter more "<<endl;
      cin>>ch1;
   }while((ch1=='y')||(ch1=='Y'));
   return 0;
}
```

Program 4: 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

Ans: #include<iostream>

```
using namespace std;
class set
{
   int **ar;
   int size;
   public:
   void setsize();
   void enter();
   int reflexive();
   void display();
   bool symmetric();
   int antisym();
   int transitive();
};

void set::setsize()
{
   cout<<"Enter the size "<<endl;
   cin>>size;
   ar=new int*[size];
```

```
    for(int i=0;i<size;i++)
    {
       ar[i]=new int[size];
    }
    for(int i=0;i<size;i++)
    {   for(int j=0;j<size;j++)
       {
          ar[i][j]=0;
       }
    }
}

void set::enter()
{
    int a,b,n;
    cout<<"Enter the no of relations"<<endl;
    cin>>n;
    if(n>(size*size))
       cout<<"Invalid no of relations"<<endl;
    else
    {
       for(int i=0;i<n;i++)
       {
          cout<<"enter the element"<<endl;
```

```
            cin>>a>>b;
            if((a==0)||(b==0)||(a>size)||(b>size))
            {
               cout<<"wrong choice"<<endl;
               exit(0);
            }
            else
            {
               ar[a-1][b-1]=1;
            }
         }
      }
}

void set::display()
{
   cout<<"The relation in matrix form is "<<endl;
   for(int i=0;i<size;i++)
   {
      for(int j=0;j<size;j++)
         cout<<ar[i][j]<<" ";
      cout<<endl;
   }
}
```

```
int set::reflexive()
{
   int flag=1;
   for(int i=0;i<size;i++)
   {
      if(ar[i][i]==0)
      {
         flag=0;
         return flag;
      }
   }
   return flag;
}

bool set::symmetric()
{
   int flag=1;
   for(int i=0;i<size;i++)
   {
      for(int j=i+1;j<size;j++)
      {
         if((ar[i][j]!=ar[j][i]))
         {
```

```
            flag=0;
            return flag;
            break;
        }
      }
   }
   return flag;
}

int set:: antisym()
{
   int flag=1;
   for(int i=0;i<size;i++)
   {
      for(int j=i+1;j<size;j++)
      {
         if(ar[i][j])
         {
            if(ar[j][i])
            {
               flag=0;
               return flag;
               break;
            }
```

```
        }
      }
   }
   return flag;
}

int set::transitive()
{
   int flag=1;
   for(int i=0;i<size;i++)
   {
      for(int j=0;j<size;j++)
      {
         if(ar[i][j]==1)
         {
            for(int k=0;k<size;k++)
            {
               if((ar[j][k]==1)&&(ar[i][k]!=1))
               flag=0;
            }
         }
      }
   }
   return  flag;
```

```
}

int main()
{
   int ch;
   char ch1;
   set a;
   a.setsize();
   a.enter();
   a.display();
   cout<<"1.equivalence"<<endl<<"2.partial
order"<<"3.none"<<"4.exit()"<<endl;
   do
   {
      cout<<"enter your choice ";
      cin>>ch;
      switch(ch)
      {
         case 1: if((a.reflexive())&&(a.symmetric())&&(a.transitive()))
                    cout<<"it is a equivalence relation"<<endl;
                 else
                    cout<<"it is not a equivalence relation"<<endl;
                 break;
         case 2: if((a.reflexive())&&(a.antisym())&&(a.transitive()))
```

```
            cout<<"it is a partial order relation"<<endl;
        else
            cout<<"it is not a partial order relation"<<endl;
        break;
    case 3: if(!a.reflexive())
            cout<<"NONE";
        else
        {
            if((!a.symmetric())||(!a.antisym()))
                cout<<"NONE";
            else
            {
                if(!a.transitive())
                    cout<<"NONE";
                else
                    cout<<"it is either equivalence or partial order";
            }
        }
        break;
    case 4:exit(0);

    default: cout<<"wrong choice"<<endl;
        break;
}
```

```
   cout<<"Do you want to enter more "<<endl;
   cin>>ch1;
   }while((ch1=='y')||(ch1=='Y'));
   return 0;
}
```

Program 5: Write a Program to generate the Fibonacci Series using recursion.

Ans:

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

int fibonacci(int n)
{
   if((n==0)||(n==1))
      return n;
   else
      return (fibonacci(n-1)+fibonacci(n-2));
}

int main()
{
   int n,i=0;
   cout<<"enter the size of fibonacci series"<<endl;
   cin>>n;
```

```
    if(n==0)
    {
       cout<<"error!!!!";
       exit(0);
    }
    else
    {
       for(int i=0;i<n;i++)
       {
          cout<<" "<<fibonacci(i);
       }
    }
    getch();
    return 0;
}
```

Program 6: Write a Program to implement Tower of Hanoi using recursion.

Ans:

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

int TOH(int n)
```

```
{
   if(n==1)
      return 1;
   else
      return (2*TOH(n-1)+1);
}

int main()
{
   int n,m;
   cout<<"\nEnter the no. of discs: ";
   cin>>n;
   m = TOH(n);
   cout<<"\nNo. of moves taken are: "<<m;
}
```

Program 7: Write a Program to implement binary search using recursion.

Ans:

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

void binarysearch(int arr[],int num,int first,int last)
{
   int mid;
```

```
    if(first>last)
        cout<<"Element not found"<<endl;
    else
    {
        mid=(first+last)/2;
        if(arr[mid]==num)
        {
            cout<<"Element is found at "<<mid+1;
        }
        else if(arr[mid]>num)
        {
            binarysearch(arr,num,first,mid-1);
        }
        else
        {
            binarysearch(arr,num,mid+1,last);
        }
    }
}

void sorting(int arr[],int n)
{
    for(int i=0;i<n-1;i++)
    {
```

```
        for(int j=0;j<n-1-i;j++)
        {
            if(arr[j]>arr[j+1])
            swap(arr[j],arr[j+1]);
        }
    }
    cout<<"the sorted array is"<<endl;
    for(int i=0;i<n;i++)
    cout<<arr[i]<<" ";
    cout<<endl;
}

void swap(int *p,int *q)
{
    int temp;
    temp=*p;
    *p=*q;
    *q=temp;
}

int main()
{
    int arr[50],beg,mid,end,e;
    int size;
```

```
   cout<<"Enter the size of array"<<endl;
   cin>>size;
   cout<<"enter the elements of array"<<endl;
   for(int i=0;i<size;i++)
   cin>>arr[i];
   sorting(arr,size);
   beg=0;
   end=size-1;
   cout<<"Enter the element to be searched"<<endl;
   cin>>e;
   binarysearch(arr,e,beg,end);
   getch();
}
```

Program 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.

Ans:

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

int bubble(short int a[],int n)
{
   int c=0;
   int temp ;
```

```
    for(int i=1;i<n;i++)
    {
        int t=0;
        for(int j=0;j<n-i; j++)
        {
            if(a[j+1]<a[j])
            {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
                c++;
                t=1;
            }
        }
        if(t==0)
        break;
        cout<<"\n";
        for(int i=0;i<n;i++)
        {
            cout<<a[i]<<"  ";
        }
    }
    return c;
}
```

```
int main()
{
   short int arr[20],n , i , arr1 ;
   cout<<"\nEnter the no of elements in array: ";
   cin>>n;
   cout<<"\nthe array is :";
   for( i=0;i<n;i++)
   {
      arr[i]=rand()%10;
      cout<<arr[i]<<" ";
   }
   cout<<"\nThe list is: \n";
   for(i=0;i<n;i++)
   {
      cout<<arr[i]<<"  ";
   }
   arr1=bubble(arr,n);

   cout<<"\nThe list after sorting is: \n";
   for(i=0;i<n;i++)
   {
      cout<<arr[i]<<"  ";
   }
```

```
    cout<<"\nThe total no. of comparisons after sorting : "<<arr1;
    return 0;
}
```

Program 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.

Ans:

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

int insertion(short int a[],int n)
{
    int key,c=0;
    for(int i=1;i<n;i++)
    {
        int t=0;
        key=a[i];
        int j=i-1;
        while(j>=0&&a[j]>key)
        {
            a[j+1]=a[j];
            j--;
            c++;
            t=1;
```

```
        }
        a[j+1]=key;
        cout<<"\n";
        for(int i=0;i<n;i++)
        {
            cout<<a[i]<<"  ";
        }
    }
    cout<<"\nThe total no. of comparisons after sorting : "<<c;
}

int main()
{
    short int arr[20],n , i , arr1 ;
    cout<<"\nEnter the no of elements in array: ";
    cin>>n;

    cout<<"\nthe array is : ";
    for( i=0;i<n;i++)
    {
        arr[i]=rand()%1000;
        cout<<arr[i]<<" ";
    }
    cout<<"\nThe list is: \n";
```

```
   for(i=0;i<n;i++)
   {
      cout<<arr[i]<<"  ";
   }
   arr1=insertion(arr,n);

   cout<<"\nThe list after sorting is: \n";
   for(i=0;i<n;i++)
   {
      cout<<arr[i]<<"  ";
   }
   return 0;
}
```

Program 10: 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).

Ans:

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

void swap(int *a,int *b)
{
   int temp;
   temp=*a;
```

```
    *a=*b;
    temp=*b;
}

void perm(int A[],int b,int n)
{
    if(b==n-1)
    {
        for(int i=0;i<n;i++)
        {
            cout<<A[i];
        }
        cout<<endl;
    }
    else
    {
        for(int i=b;i<n;i++)
        {
            swap(A[i],A[b]);
            perm(A,b+1,n);
            swap(A[i],A[b]);
        }
    }
}
```

```
void permrep(int A[],int B[],int b,int n)
{
   if(b==n)
   {
      for(int i=0;i<b;i++)
      {
         cout<<B[i];
      }
      cout<<endl;
   }
   else
   {
      for(int i=0;i<n;i++)
      {
         B[b]=A[i];
         permrep(A,B,b+1,n);
      }
   }
}

int main ()
{
   int ch;
```

```
    int A[50],B[50];
    int n,b=0;
    cout<<"Enter the size of set ";
    cin>>n;
    cout<<"Enter the elements of set"<<endl;
    for(int i=0;i<n;i++)
    {
       cin>>A[i];
    }
    cout<<"The set entered is {";
    for(int i=0;i<n;i++)
    {
       if(i==0)
       cout<<A[i];
       else
       cout<<","<<A[i];
    }
    cout<<"}";
    cout<<endl;
    cout<<"Enter your choice "<<endl<<"1.permutation with
repetition"<<endl<<"2.permutation without
repetition"<<endl<<"3.exit"<<endl;
    cin>>ch;
    switch(ch)
```

```
    {
       case 1: permrep(A,B,b,n);
                  break;
       case 2: perm(A,b,n);
                  break;
       case 3: exit(0);
       default:cout<<"wrong choice";
                  break;
    }
    getch();
}
```

Program 11: Write a Program to calculate Permutation and Combination for an input value n and r using recursive formula of nCr and nPr.

Ans:

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

int perm(int n,int r)
{
    if(r>n)
       return 0;
    else
    {
```

```
        if(r==0)
            return 1;
        else
            return(n*perm(n-1,r-1));
    }
}

int comb(int n,int r)
{
    if(r>n)
        return 0;
    else if((n==0)|(r==0)|(n==r))
        return 1;
    else
        return(comb(n-1,r-1)+comb(n-1,r));
}

int main()
{
    int n,r,ch,a,b,ch1;
    cout<<"enter the value of n"<<endl;
    cin>>n;
    cout<<"enter the value of r"<<endl;
    cin>>r;
```

```
   cout<<"enter your
choice"<<endl<<"1.permutation"<<endl<<"2.combination"<<endl<<"
3.exit"<<endl;
   cin>>ch;


   switch(ch)
   {
      case 1: a=perm(n,r);
            cout<<"permutation is "<<a<<endl;
            break;
      case 2: b=comb(n,r);
            cout<<"combination is "<<b<<endl;
            break;
      case 3: exit(0);
      default:cout<<"wrong choice";
   }
   getch();
}
```

Program 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.

Ans:

```
#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 ,r;
   cout<<"\nx1+x2+x3+---+xn=c";
   cout<<"\nEnter the no of variables (n) : ";
   cin>>n;

   cout<<"\nEnter the value of total sum (c<=10) : ";
   cin>>r;

   cout<<"\nNUMBER OF POSSIBLE SOLUTIONS OF THE GIVEN
EQUATION IS : ";
   cout<<comb(n+r-1,r);
}
```

Program 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-conditional

Ans:

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

int main()
{
   int x[4],y[4],dis[4],con[4],NOR[4],NAND[4],Cond[4],Bicond[4],Negx[4],Negy[4];
   char ch;
   do
   {
      for(int i=0;i<4;i++)
      {
         cout<<"Enter the"<<" "<<(i+1)<<" "<<"value of x and y"<<endl;
         cin>>x[i]>>y[i];
         dis[i]=x[i]|y[i];
         con[i]=x[i]&y[i];
         NOR[i]=!dis[i];
         NAND[i]=!con[i];
```

```
        Cond[i]=!x[i]|y[i];
        Bicond[i]=((!x[i]|y[i])&(!y[i]|x[i]));
        Negx[i]=!x[i];
        Negy[i]=!y[i];
      }

      cout<<"x |y  |or
|and|NOR|NAND|Cond|Bicond|Negx|Negy"<<endl;
      for(int i=0;i<4;i++)
      {
        cout<<x[i]<<" | "<<y[i]<<" | "<<dis[i]<<" | "<<con[i]<<" |
"<<NOR[i]<<" | "<<NAND[i]<<"  | ";
        cout<<Cond[i]<<"  |"<<Bicond[i]<<"    |"<<Negx[i]<<"
|"<<Negy[i]<<endl;
      }

      cout<<"do you want to continue(y/n)\n";
      cin>>ch;
   }while(ch=='y');

   return 0;
}
```

Program 14: Write a program to accept an input n from the user and graphically represent the values of T (n) where n varies from 0 to n

for the recurrence relations. For e.g. T(n) = T(n-1) + n, T(0) = 1, T(n) = T(n-1) + n^2, T(0) =1, T(n) = 2*T(n)/2 + n, T(1)=1.

Ans:

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

int Ta(int n)
{
   if(n==0)
      return 1;
   else
      return (n+Ta(n-1));
}

int Tb(int n)
{
   if(n==0)
      return 1;
   else
      return (pow(n,2)+Tb(n-1));
}

int Tc(int n)
{
```

```
    if(n==0)
        return 1;
    else
        return (n+(2*Tc(n/2)));
}

int main()
{
    int n,c;

    cout<<"VALUES OF RECURRENCE RELATIONS :"<<endl<<endl;
    cout<<"\t 1. T(n)= T(n-1)+n       ,T(0)=1 "<<endl;
    cout<<"\t 2. T(n)= T(n-1)+n^2     ,T(0)=1 "<<endl;
    cout<<"\t 3. T(n)= 2*T(n/2)+n     ,T(1)=1 "<<endl;

    cout<<"\nEnter your choice(between 1,2,3) :     ";
    cin>>c;
    if((c>3)||(c<=0))
        cout<<"\nInvalid choice!!";
    cout<<"\nEnter the value for n : ";
    cin>>n;

    if(c==1)
    {
```

```
        int r=Ta(n);
        cout<<"\nResult : "<<r<<endl;
    }

    else if(c==2)
    {
        int r=Tb(n);
        cout<<"\nResult : "<<r<<endl;
    }

    else if(c==3)
    {
        int r=Tc(n);
        cout<<"\nResult : "<<r<<endl;
    }

    return 0;
}
```

Program 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)).

Ans:

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

```
int main()
{
   int arr[20],deg,x,sum=0;
   char ch;
   do
   {
      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;
      cout<<"\n Enter the value of x : ";
```

```
        cin>>x;
        for(int i=deg;i>=0;i--)
        {
           sum+=(arr[i]*pow(x,i));
        }
        cout<<"\nThe solution of this polynomial is :"<<sum;
        char ch;
        cout<<"\n Do you want to continue?(y/n):";
        cin>>ch;

    }while(ch=='y'||ch=='Y');
}
```

Program 16: Write a Program to represent Graphs using the Adjacency Matrices and check if it is a complete graph.

Ans:

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

int main()
{
    char choice;
    int v,flag=0,q;
    cout<<"enter the number of vertices";
    cin>>v;
    int ar[v][v];
```

```
   for(int i=0;i<v;i++)
   {
      for(int j=0;j<v;j++)
      {
         cout<<"\n How many edge from "<<(char)(97+i)<<" to
"<<(char)(97+j)<<" - ";
         cin>>ar[i][j];
      }
   }

   cout<<"the adjancy matrix of graph is \n";
   for(int k=0;k<v;k++)
   {
      cout<<endl;
      for(int l=0;l<v;l++)
      cout<<ar[k][l]<<" ";
   }

   for(int p=0;p<v;p++)
   {
      cout<<endl;
      for(int q=0;q<v;q++)
      {
         if((p!=q) && (p<q))
```

```
        {
            if(ar[p][q]!=1)
            {
                flag=1;
                break;
            }
        }
    }
}

if(flag==1)
{
    cout<<"this is a not complete graph\n";
}
else
{
    cout<<"this is a complete graph\n";
}
}
```

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

Ans:

```
#include<iostream>
using namespace std;
int main()
```

```
{
   int arr[50][50];
   int count=0;
   int out=0;
   int in=0;
   int i,j;
   char ch;
   int v,e;

   cout<<"\n Enter the number of vertex : "<<endl;
   cin>>v;

   cout<<"\n Enter the edges for the graph : "<<endl;
   for(int i=0;i<v;i++)
   {
      for(int j=0;j<v;j++)
      {
         cout<<"\n Enter the no. of edges ";
         cout<<"from"<<" "<<(char)(i+97)<<" "<<"to vertex"<<"
"<<(char)(j+97)<<"  : ";
         cin>>e;
         if(e>0)
         {
            arr[i][j]=e;
```

```
        }
        else
            arr[i][j]=0;
    }
}

cout<<"\n The matrix you entered is : "<<endl;
for( i=0;i<v;i++)
{
    for(j=0;j<v;j++)
    {
        cout<<arr[i][j]<<" ";
    }
    cout<<endl;
}

for( i=0;i<v;i++)
{
    for(j=0;j<v;j++)
    {
        if(arr[i][j]>0)
        {
            out=out+arr[i][j];
        }
```

```
    }
    cout<<"\n The out degree of the vertex is"<<(char)(i+97)<<out;
    out=0;
  }

  for(int i=0;i<v;i++)
  {
    for(int j=0;j<v;j++)
    {
    if(arr[j][i]>0)
      {
        in=in+arr[j][i];
      }
    }
    cout<<"\n The in degree of the vertex is"<<(char)(i+97)<<in;
    in=0;
  }
}
```

Program 18: 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.

Ans:

```
#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;
    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;
}
```

Program 18: 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.

Ans:

```
#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;
    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;
}
```

Program 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).

Ans: #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;
}
```

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

Ans:

```
#include<iostream>
using namespace std;
int main()
{
    int m,l,i;
```

```
    cout<<"enter the degree of tree:";
    cin>>m;

    cout<<"enter the value of internal vertices:";
    cin>>i;

    l=i*(m-1)+1;

    cout<<"the number of leaves:"<<l<<endl;
}
```