Question 1.

/*Implement a class Complex which represents the Complex Number data type. Implement

the following operations:

1. Constructor (including a default constructor which creates the complex number 0+0i).

2. Overloaded operator+ to add two complex numbers.

3. Overloaded operator* to multiply two complex numbers.

4. Overloaded << and >> to print and read Complex Numbers.*/

```
#include<iostream>
using namespace std;
class complex
{
        float x;
        float y;
public:
    complex()
        {
            x=0;
            y=0;
        }

        complex operator+(complex);
        complex operator*(complex);
        friend istream &operator >>(istream &input,complex &t)
        {
            cout<<"Enter the real part";
            input>>t.x;
            cout<<"Enter the imaginary part";
            input>>t.y;
```

```
        }
        friend ostream &operator <<(ostream &output,complex &t)
        {
                        output<<t.x<<"+"<<t.y<<"i\n";
        }

};

complex complex::operator+(complex c)
{
        complex temp;
        temp.x=x+c.x;
        temp.y=y+c.y;
        return(temp);
}


complex complex::operator*(complex c)
{
        complex temp2;
        temp2.x=(x*c.x)-(y*c.y);
        temp2.y=(y*c.x)+(x*c.y);
        return (temp2);
}


int main()
{
        complex c1,c2,c3,c4;
    cout<<"Default constructor value=\n";
    cout<<c1;
```

```
        cout<<"\nEnter the 1st number\n";
        cin>>c1;
        cout<<"\nEnter the 2nd number\n";
        cin>>c2;
        c3=c1+c2;
        c4=c1*c2;
        cout<<"\nThe first number is ";
        cout<<c1;
        cout<<"\nThe second number is ";
        cout<<c2;
        cout<<"\nThe addition is ";
        cout<<c3;
        cout<<"\nThe multiplication is ";
        cout<<c4;
        return 0;
}
```

==========

Question 2

/*Implement a class Quadratic that represents two degree polynomials i.e., polynomials of type ax2+bx+c.

Your class will require three data member corresponding to a, b and c.

Implement the following operations:

1.A constructor(including a default constructor which creates the 0 polynomial).

2.Overloaded operator+ to add two polynomials of degree 2.

3.Overloaded << and >> to print and read polynomials.

To do this, you will need to decide what you want your input and output format to look like.

4.A function eval that computes the value of a polynomial for a given value of x.

5.A function that computes the two solutions of the equation ax 2 +bx+c=0

***/**

```
#include<iostream>
#include<cmath>
using namespace std;
class polynomial
{ int a,b,c;
 public:
 polynomial() //default constructor
 {
 a=0;
 b=0;
 c=0;

      }
      polynomial(int x,int y,int z) //parameterized constructor
      { a=x;b=y;c=z;

      }

 friend istream &operator >> (istream &IN, polynomial &T) //operator >> overloaded
 {
  IN>>T.a>>T.b>>T.c;
  return IN;
 }
```

```
friend ostream &operator << (ostream &OUT, polynomial &T) //operator << overloaded
 {
  OUT<<T.a<<"x^2+"<<T.b<<"x+"<<T.c;
  return OUT;
 }

polynomial operator +(polynomial T) //opeartor + overloaded
{
polynomial R;
R.a=a+T.a;
R.b=b+T.b;
R.c=c+T.c;
return R;
}
void eval(polynomial T,int x) //eval fn to evaluate the polynomial for given value of x
     {int z;
      z=T.a*x*x+T.b*x+T.c;
      cout<<" for x ="<<x<<" is :"<<z<<"\n";
     }

     void compute(polynomial T) //compute fn to find roots of polynomial
     {
     float x,y1,y2;
     x= T.b*T.b-4*T.a*T.c;
     if(x>0)
     {
     cout<<"Roots are real & not equal\n";
     y1 = (-T.b+sqrt(x))/(2*T.a);
     y2 = (-T.b-sqrt(x))/(2*T.a);
     cout<<y1<<"\n";
     cout<<y2<<"\n";
```

```
    }
    else if(x==0)
    {cout<<"roots are real & equal\n";
    y1= -T.a/(2*T.a);
    cout<<y1<<"\n";
    }
    else if(x<0)
    {
    cout<<"complex Roots\n";

    }


    }
};
int main()
{
int x,ch;
char p,P;
  polynomial s1(5,6,10),s2,s3;
                do
                {
                cout<<"######################MENU###################\n";
                cout<<"1]display default polynomial\n";
                cout<<"2]accept & display polynomial\n";
                cout<<"3]add two polynomial\n";
                cout<<"4]find f(x) for given x\n";
                cout<<"5]find roots of give polynomial\n";
                cout<<"enter your choise\n";
                cin>>ch;
                switch(ch)
```

```
                {
                case 1:
                cout<<"display stored polynomial\n";
cout<<s1<<endl;
                break;

                case 2:
cout<<"accept & display\n";
cout<<"enter the  polynomial :";
cin>>s2;
cout<<s2<<endl;
                break;

                case 3:
                cout<<"addition of two polynomials\n";
                cout<<"enter the 1st polynomial :";
cin>>s1;
                cout<<s1<<endl;
                cout<<"enter the 2nd polynoimial:";
                cin>>s2;
                cout<<s2<<endl;
                s3=s1+s2;
                cout<<"addition is:";
cout<<s3<<endl;
                break;

                case 4:
                cout<<"calculate f(X) fo hiven x\n";
                cout<<"enter the polynomial :";
cin>>s1;
cout<<"enter the value for x = ";
```

```
  cin>>x;
  s2.eval(s1,x);
                  break;

                  case 5:
                  cout<<"find roots of polynomial\n";
                  cout<<"enter the polynomial :";
  cin>>s2;
                  s2.compute(s2);
                  break;
                  }
                  cout<<"do you want to continue(y/n)";
                  cin>>p;

                 }while(p=='y'||P=='Y');
           cout<<"goodby have a nice day\n";
          return 0;
}
```

==========

Question 3

/*Implement a class CppArray which is identical to a one-dimensional C++ array (i.e., the index set is a set of consecutive integers starting at 0) except for the following :

1. It performs range checking.

2. It allows one to be assigned to another array through the use of the assignment operator (e.g. cp1= cp2)

3. It supports a function that returns the size of the array.

4. It allows the reading or printing of array through the use of cout and cin.

***/**

```
#include<iostream>
using namespace std;
class CppArray
{
  private:
  int a[10],i,j,n,b[10],temp;
  public:
  void get();
  void print();
  void sort();
  void range();
  void exchange();
  int size();
};
void CppArray::get()
{
  cout<<"\n Enter the limit of array ";
  cin>>n;
  cout<<"\n Enter the array elements ";
  for(i=0;i<n;i++)
  {
     cout<<"\n a["<<i<<"]=";
     cin>>a[i];
  }
}
void CppArray::print()
{
  cout<<"\n The entered array is ";
```

```
  for(i=0;i<n;i++)
  {
     cout<<"\n a["<<i<<"]="<<a[i];
  }
}
void CppArray::sort()
{
  for(j=0;j<n;j++)
  {
    for(i=0;i<n-1;i++)
    {
      if(a[i]>a[i+1])
      {
        temp=a[i+1];
        a[i+1]=a[i];
        a[i]=temp;
      }
    }
  }
  cout<<"\n The sorted array is ";
  for(i=0;i<n;i++)
  {
     cout<<"\n a["<<i<<"]="<<a[i];
  }
}
void CppArray::range()
{
  cout<<"\n The range of array is from "<<a[0]<<" to "<<a[n-1];
}
void CppArray::exchange()
{
```

```
  for(i=0;i<n;i++)
  {
     b[i]=a[i];
  }
  cout<<"\n The exchanged array is ";
  for(i=0;i<n;i++)
  {
     cout<<"\n b["<<i<<"]="<<b[i];
  }
}
int CppArray::size()
{
  return n;
}
int main()
{
  int a;
  CppArray obj;
  obj.get();
  obj.print();
  obj.range();
  obj.exchange();
  obj.sort();
  a=obj.size();
  cout<<"\n The size of array is "<<a;
  return 0;
}
```

==========

Question 4.

/*Write a C++ program create a calculator for an arithmetic operator (+, -, *, /). The program should take two operands from user and performs the operation on those two operands depending upon the operator entered by user. Use a switch statement to select the operation. Finally, display the result. Some sample interaction with the program might look like this:

Enter first number, operator, second number: 10 / 3

Answer = 3.333333

Do another (y/n)? y

Enter first number, operator, second number: 12 + 100

Answer = 112

Do another (y/n)? n

***/**

```
#include<iostream>
using namespace std;
class Calculator
{
  private:
        float num1,num2,result;
        char op;
  public:
        void get();
        void calculate();
};
void Calculator::get()
{
  cout<<"\nEnter first number, operator, second number:\n";
  cin>>num1;
  cin>>op;
  cin>>num2;
```

```
}
void Calculator::calculate()
{

  switch(op)
  {
    case '+':
                    result=num1+num2;
            cout<<" Answer = "<<result;

                    break;
    case '-':
                      result=num1-num2;
            cout<<" Answer = "<<result;

                    break;
    case '*':
                     result=num1*num2;
            cout<<" Answer = "<<result;

                     break;
    case '/':
                    if(num2==0)
                    cout<<"\n Error. Not valid.";
                     result=num1/num2;
            cout<<" Answer = "<<result;

    break;
  }

}
```

```
int main()
{
  char ag;
  Calculator obj;
  x:obj.get();
  obj.calculate();
  cout<<"\n Do another (y/n)? ";
  cin>>ag;
  if(ag=='y'||ag=='Y')
  goto x;
  return 0;
}
```

==========

Question 5.

/*Develop an object oriented program in C++ to create a database of student information system containing the following information: Name, Roll number, Class, division, Date of Birth, Blood group, Contact address, telephone number, driving license no. etc Construct the database with suitable member functions for initializing and destroying the data viz constructor, default constructor, Copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.*/

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

```
using namespace std;

class person_additional_info
{
	char address[20],license[20],insurance[20];
	long int contact;

	public:
		person_additional_info() //Default constructor
		{
			strcpy(address,"XYZ");
				strcpy(license,"XY-0000000000");
			strcpy(insurance,"XY00000000X");
			contact=000000000;

		}

	~person_additional_info() //Destructor
	 {
	       cout<<"I am in Destructor";
	 }

	    friend class person;  // Declaration Friend class
};

//Definition of friend class
class person
{
	char name[20], dob[10], blood[10];
	float ht,wt;
	static int count; // Static variable
```

```
person_additional_info *pai;


public:
   person()  //Default constructor
     {
       strcpy(name,"XYZ");
       strcpy(dob,"dd/mm/yy");
       strcpy(blood,"A +");
       ht=0;
       wt=0;
    pai=new person_additional_info;
 }
          person(person*p1) //Copy constructor
          {
                strcpy(name,p1->name);
                strcpy(dob,p1->dob);
                strcpy(blood,p1->blood);
                ht=p1->ht;
                wt=p1->wt;
            pai=new person_additional_info;
          strcpy(pai->address,p1->pai->address);
              strcpy(pai->license,p1->pai->license);
              strcpy(pai->insurance,p1->pai->insurance);
          pai->contact=p1->pai->contact;
 }
          static void showcount()  //Static member function
          {
                    cout<<"\nNo of records count="<<count<<"\n";
 }
~person()    //Destructor
 {
```

```
        cout<<"\nI am in Destructor\n";
    }
        void getdata(char name[20]);
        inline void display();   // Inline function declaration

};
void person::getdata(char name[20])
{
    strcpy(this->name,name);     //this pointer
    cout<<"\n Enter date of birth";
    cin>>dob;
    cout<<"\n Enter blood group";
      cin>>blood;
    cout<<"\n Enter height";
      cin>>ht;
    cout<<"\n Enter weight";
      cin>>wt;
    cout<<"\n Enter address";
    cin>>pai->address;
    cout<<"\n Enter Licence number";
      cin>>pai->license;
    cout<<"\n Enter Insurance policy number";
      cin>>pai->insurance;
    cout<<"\n Enter Contact number";
    cin>>pai->contact;
    count++;
}
//inline function definition
void person::display()
{
    cout<<"\t"<<name;
```

```
        cout<<"\t"<<dob;
        cout<<"\t"<<blood;
        cout<<"\t"<<ht;
        cout<<"\t"<<wt;
        cout<<"\t"<<pai->address;
        cout<<"\t"<<pai->license;
        cout<<"\t"<<pai->insurance;
        cout<<"\t"<<pai->contact;
}

int person::count;  //Static variable definition

int main()
{
        person *p1,*p2;
        int ch;
        p1=new person;  //call default constructor & dynamic memory allocation
        p2=new person(p1); //call copy constructor
        cout<<"\tName";
        cout<<"\tDob";
        cout<<"\t   Blood";
        cout<<"\tHt";
        cout<<"\tWt";
        cout<<"\tAddress";
        cout<<"\tLicense";
        cout<<"\tInsurance";
        cout<<"\tContact";
        cout<<endl;
        cout<<"Default Constructor Value \n";
        p1->display();
        cout<<"\n";
```

```
cout<<"Copy Constructor Value \n";
p2->display();
   int n;
cout<<"\nEnter how many records you want??";
cin>>n;
person p3[n];     //array of object
char name[20];
int x=0;
do
{
 cout<<"\nWelcome to Personal database system";
            cout<<"\n1.Enter the record";
            cout<<"\n2.Display the record";
            cout<<"\n3.Exit";
            cin>>ch;
            switch(ch)
            {
                        case 1:
                        {
                                    cout<<"\nEnter the Name ";
                                    cin>>name;
                                    p3[x].getdata(name);
                                    person::showcount();  // calls static function
                                    x++;
                        break;
                        }
                        case 2:
                        {
                                    cout<<"\tName";
                                    cout<<"\tDob";
                                    cout<<"\t   Blood";
```

```
                                cout<<"\tHt";
                                cout<<"\tWt";
                                cout<<"\tAddress";
                                cout<<"\tLicense";
                                cout<<"\tInsurance";
                                cout<<"\tContact";
                                for(int i=0;i<n;i++)
                                {
                                        cout<<"\n";
                                        p3[i].display(); //calls inline function
                                }
                                break;
                        }
                }
        }while(ch!=3);
        delete p1; //dynamic memory de-allocation
        delete p2;
        return 0;

}
```

==========

Question 6.

/* : Implement C++ program to write a class template to represent a generic vector. Include following member functions:

To create the vector.

To modify the value of a given element

To multiply by a scalar value

To display the vector in the form (10,20,30,...)*/

```
#include<iostream> //header file
#include<vector>
using namespace std; //declared scope of program
void display (vector<int>&v) //function declaration
{
	for(int i=0;i<v.size();i++)
	{
		cout<<"   "<<v[i];
	}
	cout<<" \n";
}
int main()
{
	vector<int> v;
	cout<<"\n Initial Size: "<<v.size();
	int x;
	cout<<"\n Enter The 5 Element: ";
	for(int i=0;i<5;i++)
	{
		cin>>x;
		v.push_back(x);
	}
	cout<<"\n Size After Insertion: "<<v.size();
	cout<<"\n Vector Element: ";
	display(v);
	cout<<"\nVector Element After insertion of 3 at End of Vector: ";
	v.push_back(3);
	display(v);
	vector<int>::iterator itr=v.begin();
```

```
        itr=itr+3;
        v.insert(itr,9);
        cout<<"\n Content After Insertion 9 at position 3rd: ";
        display(v);
        v.erase(v.begin()+3,v.begin()+5);
        cout<<"\n After Erasing(3rd-4th position): ";
        display(v);
}
```

==========

Question 7.

**/*
Create a class Rational Number (fractions) with the following capabilities:
a) Create a constructor that prevents a 0 denominator in a fraction, reduces or simplifies fractions that are not in reduced form and avoids negative denominators.
b) Overload the addition, subtraction, multiplication and division operators for this class.
c) Overload the relational and equality operators for this class.
*/**

```
#include<iostream>
#include<cstdlib>
using namespace std;
class Rational1
{
 int numerator,denominator;
public:
```

```
 Rational1();
 Rational1(int,int);
 Rational1 operator+(Rational1);
 Rational1 operator-(Rational1);
 Rational1 operator*(Rational1);
 Rational1 operator/(Rational1);
 bool operator>(Rational1);
 bool operator<(Rational1);
 bool operator>=(Rational1);
 bool operator<=(Rational1);
 bool operator==(Rational1);
 bool operator!=(Rational1);
 void printRational1();
 void reduce();
};

// default constructor: parameters are numerator and denominator respectively
Rational1::Rational1(int n, int d) {
 numerator = n;
 denominator = d;
 if (d==0 || d<0)
 {
        cout<<"Cannot enter zero or negative numbers for denominator"<<endl;
        exit(1);
 }
 reduce();
}
Rational1::Rational1() {
 numerator = 0;
 denominator =1;
}
```

```
//--------------------------------- add -----------------------------------
// overloaded +: addition of 2 Rational1s, current object and parameter
Rational1 Rational1::operator+(Rational1 a) {
 Rational1 t;
 t.numerator = a.numerator * denominator + a.denominator * numerator;
 t.denominator = a.denominator * denominator;
 t.reduce();
 return t;
}

//-------------------------------- subtract ----------------------------------
// subtraction of 2 Rational1s, current object and parameter

Rational1 Rational1::operator-(Rational1 s) {
 Rational1 t;
 t.numerator = s.denominator * numerator - denominator * s.numerator;
 t.denominator = s.denominator * denominator;
 t.reduce();
 return t;
}

//------------------------------- multiply -----------------------------------
// multiplication of 2 Rational1s, current object and parameter
Rational1 Rational1::operator*(Rational1 m) {
 Rational1 t;

 t.numerator = m.numerator * numerator;
 t.denominator = m.denominator * denominator;
 t.reduce();
 return t;
```

```
}

//-------------------------------- divide -------------------------------------
// division of 2 Rational1s, current object and parameter,
// division by zero crashes
Rational1 Rational1::operator/(Rational1 v) {
 Rational1 t;

 t.numerator = v.denominator * numerator;
 t.denominator = denominator * v.numerator;
 t.reduce();

 return t;
}
bool Rational1::operator>(Rational1 v) {
 float f1=numerator/float(denominator);
 float f2=v.numerator/float(v.denominator);
 if(f1>f2)
 return true;
 else return false;
}
bool Rational1::operator<(Rational1 v) {
 float f1=numerator/float(denominator);
 float f2=v.numerator/float(v.denominator);
 if(f1<f2)
 return true;
 else return false;

}
bool Rational1::operator>=(Rational1 v) {
 float f1=numerator/float(denominator);
```

```
 float f2=v.numerator/float(v.denominator);
 if(f1>=f2)
 return true;
 else return false;

}
bool Rational1::operator<=(Rational1 v) {
 float f1=numerator/float(denominator);
 float f2=v.numerator/float(v.denominator);
 if(f1<=f2)
 return true;
 else
 return false;

}
bool Rational1::operator==(Rational1 v) {
 float f1=numerator/float(denominator);
 float f2=v.numerator/float(v.denominator);
 if(f1==f2)
 return true;
 else
 return false;
}
bool Rational1::operator!=(Rational1 v) {
 float f1=numerator/float(denominator);
 float f2=v.numerator/float(v.denominator);
 if(f1!=f2)
 return true;
 else return false;
}
//---------------------------- printRational1 ----------------------------------
```

```
void Rational1::printRational1() {
 if (denominator == 0)
 cout << endl << "DIVIDE BY ZERO ERROR!!!" << endl;
 else if (numerator == 0)
 cout << 0;
 else
 cout << numerator << "/" << denominator;
}

//--------------------------------- reduce ------------------------------------
// reduce fraction to lowest terms
void Rational1::reduce() {
 int n = numerator < 0 ? -numerator : numerator;
 int d = denominator;
 int largest = n > d ? n : d;
 int gcd = 0; // greatest common divisor

 for (int loop = largest; loop >= 2; loop--)
 if (numerator % loop == 0 && denominator % loop == 0) {
 gcd = loop;
 break;
 }

 if (gcd != 0) {
 numerator /= gcd;
 denominator /= gcd;
 }
}
int main()
{
Rational1 r1,r2,r3;
```

```
int n,d;
int choice;
char ch;
cout<<"Enter in format :numerator <space> denominator\n";
cout<<"Enter first Rational1 number - numerator and denominator\t: ";
cin>>n>>d;
r1=Rational1(n,d);
cout<<"Enter second Rational1 number - numerator and denominator\t:";
cin>>n>>d;
r2=Rational1(n,d);
do
{
 cout<<"**********MENU**********\n";
 cout<<"1. +\t2. -\t3.*\t4. /\t5. >\t6. <\n7. >=\t8. <=\t9. !=\t10. ==";
 cout<<"\n11. Print\t12.Exit\n";
 cout<<"Enter your choice";
 cin>>choice;
 switch(choice)
 {
 case 1:
 r3=r1+r2;
 r3.printRational1();
 break;
 case 2:
 r3=r1-r2;
 r3.printRational1();
 break;
 case 3:
 r3=r1*r2;
 r3.printRational1();
 break;
```

```
case 4:
r3=r1/r2;
r3.printRational1();
break;
case 5:
if(r1>r2)
{
r1.printRational1();
cout<<" is greater than ";
r2.printRational1();
}
else
{
r1.printRational1();
cout<<" is not greater than ";
r2.printRational1();
}
break;
case 6:
if(r1<r2)
{
r2.printRational1();
cout<<" is greater than ";
r1.printRational1();
}
else
{
r2.printRational1();
cout<<" is not greater than ";
r1.printRational1();
}
```

```
break;
case 7:
if(r1>=r2)
{
r1.printRational1();
cout<<" is greater than or equal to ";
r2.printRational1();
}
else
{
r1.printRational1();
cout<<" is not greater than or equal to ";
r2.printRational1();
}
break;
case 8:
if(r1<=r2)
{
r2.printRational1();
cout<<" is greater than or equal to ";
r1.printRational1();
}
else
{
r2.printRational1();
cout<<" is not greater than or equal to ";
r1.printRational1();
}
break;
case 9:
if(r1!=r2)
```

```
{
r1.printRational1();
cout<<" is not equal to ";
r2.printRational1();
}
else
{
r1.printRational1();
cout<<" is  equal to ";
r2.printRational1();
}
break;
case 10:
if(r1==r2)
{
r1.printRational1();
cout<<" is equal to ";
r2.printRational1();
}
else
{
r1.printRational1();
cout<<" is not equal to ";
r2.printRational1();
}
break;
case 11:
cout<<"\nFirst Rational Number\n";
r1.printRational1();
cout<<"\nSecond Rational Number\n";
r2.printRational1();
```

```
 cout<<"\nThird Rational Number\n";
 r3.printRational1();
 break;
 case 12:
 exit(0);;
 }
cout<<"\nDo you want to continue(y/n)\n";
cin>>ch;
}while(ch=='y' || ch=='Y');
return 0;
}
```

Question 8.

/*

Imagine a publishing company which does marketing for book and audio cassette versions. Create a class publication that stores the title (a string) and price (type float) of publications.

From this class derive two classes: book which adds a page count (type int) and tape which adds a playing time in minutes (type float).

Write a program that instantiates the book and tape class, allows user to enter data and displays the data members. If an exception is caught, replace all the data member values with zero values

***/**

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

```
using namespace std;
//base class publication
class publication
{
private:
string title;
float prices;
public:
publication()
 {
title="";
prices=0.0;
}
void get_data()
{
cout<<"\nEnter Title :";
cin.ignore();//clear input buffer
getline(cin,title);
cout<<"\nEnter Price : ";
cin>>prices;
}
void put_data()
{
cout<<"\n _____ \n";
cout<<"\n Information : " <<endl;
cout<<"\n Title :"<<title;
cout<<"\n Price :"<<prices;
}
};
class book: public publication
{
```

```
private:
int pages;
public:
book(){
pages=0;
}
void get_data()
{
publication::get_data();
cout<<endl;
cout<<"Enter Page Count : \n";
cin>>pages;
}
void put_data()
{
publication::put_data();
try{
if(pages<0)
 throw pages;}
 catch(int f)
 {
 cout<<"\n error: pages not valid :"<<f;
 pages=0;
 }
cout<<"\n Pages Are :"<<pages;
}
};
class tape: public publication
{
private:
float playtime;
```

```
public:
tape()
{
playtime=0.0;
}
void get_data()
{
publication::get_data();
cout<<"Enter Play Time Of Cassette \n";
cin>>playtime;
}
void put_data()
{
publication::put_data();
try
{
if(playtime<0.0)
throw playtime;
}
catch(float r)
{
cout<<"\n Error: Invalid Playtime : "<<playtime;
playtime=0.0;
}
cout<<"\n Playtime is : "<<playtime;
}
};
int main()//main program
{
book b[10];// arrray of objects
tape t[10];
```

```
int choice=0,bookCount=0,tapeCount=0;
cout<<"-----------------------";
do
{
cout<<"\n 1. Add book ";
cout<<"\n 2. Add tape: ";
cout<<"\n 3. Display book ";
cout<<"\n 4. Display tape";
cout<<"\n 5. Exit:"<<endl;
cout<<"\n Enter Choice : ";
cin>>choice;
switch(choice)
{
case 1:
 {
 cout<<"\n--------------\n";
 cout<<"Add Book: \n";
 b[bookCount].get_data();
 bookCount++;
 break;
 }
case 2:
 {
 cout<<"\n--------------\n";
 cout<<"Add Tape: \n";
 t[tapeCount].get_data();
 tapeCount++;
 break;
 }
case 3:
 {
```

```
 cout<<"\n (books)";
 for(int j=0;j<bookCount;j++)
 {
 b[j].put_data();
 }
 break;
 }
case 4:
 {
 cout<<"\n (tape)";
 for(int j=0;j<tapeCount;j++)
 {
 t[j].put_data();
 }
 break;
 }
case 5:
 {
 cout<<"**********Program Exited Successfully**********"<<endl;

exit(0);

 }
 default:
 {
 cout<<"\n Invalid";
 }
}
}
while(choice!=5);
return 0;
```

```
}
```

==========

Question 9.

```
/*
Write a function in C++ to count and display the number of lines not starting with alphabet 'A' present in a text file "STORY.TXT".
October 19, 2018
 Write a function in C++ to count and display the number of lines not starting with
alphabet 'A' present in a text file "STORY.TXT".
Example:
If the file "STORY.TXT" contains the following lines,
The roses are red.
A girl is playing there.
There is a playground.
An aeroplane is in the sky.
Numbers are not allowed in the password.
The function should display the output as 3.
*/


#include<iostream>
#include<fstream>
using namespace std;
int main()
{
   ifstream fin;
```

```
    fin.open("Story.txt");

    char str[100];
    int count=0;
    int count1=0;

    while(!fin.eof())
    {
        fin.getline(str,100);
        if(str[0]!='A')
        {
            count++;
        }
        else if(str[0]=='A'&&str[1]==' ')
        {
            count1++;
        }
    }

    cout<<"The number of lines not starting with 'A' are:"<<count<<"\n";
    cout<<"The number of lines starting with 'A' are:"<<count1<<"\n";

    fin.close();
    return 0;
}
```

==========
==========

Question 10.

/*

Write C++ Program with base class convert declares two variables, val1 and val2,

which hold the initial and converted values, respectively. It also defines the functions

getinit( ) and getconv( ), which return the initial value and the converted value.

These elements of convert are fixed and applicable to all derived classes that will inherit convert. H

owever, the function that will actually perform the conversion, compute( ),

is a pure virtual function that must be defined by the classes derived from convert.

The specific nature of compute( ) will be determined by what type of conversion is taking place.

***/**

```
#include <iostream>

using namespace std;


class convert {
    protected:
        double val1;
        double val2;
    public:
    convert(double i){
    val1=i;
}
double getconv()  {return val2;}
```

```
double getinit()  {return val1;}
virtual void compute ()=0;
};
  //Liters to gallons
class l_to_g:public convert
{
public:
l_to_g(double i):convert(i){}
void compute()
{
val2=val1/3.7854;
}
};


class f_to_c:public convert
{
public:
f_to_c(double i):convert(i){}
void compute()
{
val2=(val1-32)/1.8;
}
};


class c_to_k:public convert
{
public:
c_to_k(double i):convert(i){}
void compute()
```

```
{
val2=val1+273.15;
}
};


class k_to_f:public convert
{
public:
k_to_f(double i):convert(i){}
void compute()
{
val2=val1*915-459.67;
}
};
int main()
{
convert *p;
l_to_g ob(5);
f_to_c ob1(70);
c_to_k ob2(50);
k_to_f ob3(50);
p=&ob;
cout<<p->getinit()<<"Liters is";
p->compute();
cout<<p->getconv()<<"gallons\n";
p=&ob1;
cout<<p->getinit()<<"Farenheit is";
p->compute();
cout<<p->getconv()<<"Celcius\n";
p=&ob2;
```

```
cout<<p->getinit()<<" Celcius is";
p->compute();
cout<<p->getconv()<<"kelvin\n";
p=&ob3;
cout<<p->getinit()<<" kelvin is";
p->compute();
cout<<p->getconv()<<"Farenheit\n";
return 0;
}
```

==========