1. Write a C++ program to create array of objects of Book (id, title, author, price, publication) class. Read number of objects n from the user. **(15 marks)**

**#include<iostream>**

**#include <string>**

**using namespace std;**

**class Book**

**{**

**int id;**

**string title;**

**string author;**

**float price;**

**string publication;**

**public:**

**void getdata();//Declaration of function**

**void putdata();//Declaration of function**

**};**

**void Book::getdata(){//Defining of function**

**cout<<"Enter Id : ";**

**cin>>id;**

**cout<<"Enter title : ";**

**cin>>title;**

**cout<<"Enter author : ";**

**cin>>author;**

**cout<<"Enter price : ";**

**cin>>price;**

**cout<<"Enter publication : ";**

**cin>>publication;**

**}**

**void Book::putdata(){//Defining of function**

**cout<<"id="<<id<<endl;**

**cout<<"title="<<title<<endl;**

**cout<<"author="<<author<<endl;**

**cout<<"price="<<price<<endl;**

**cout<<"publication="<<publication<<endl;**

**cout<<endl;**

**}**

**int main(){**

**//One member**

**int n;**

**cout<<"Enter the number of books:";**

**cin>>n;**

**Book emp[n];**

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

**{**

**emp[i].getdata();//Accessing the function**

**}**

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

**{**

**emp[i].putdata();//Accessing the function**

**}**

**return 0;**

**}**

1. Create a class Cone(radius, height). Declare a static member count that will store the number of objects created. Write a static function that display() the number of objects created. Write an area() function as inline function to display area of the cone. (15 marks)

#include <iostream>

#include <cmath>

using namespace std;

class Cone

{

private:

    double radius;

    double height;

    static int count; // static member variable to store the number of objects created

public:

    // Constructor to initialize radius and height of the cone

    Cone(double r, double h)

    {

        radius = r;

        height = h;

        ++count; // increment the count each time an object is created

    }

    // static function to display the number of objects created

    static void display()

    {

        cout << "Number of objects created: " << count << endl;

    }

    // inline function to calculate and display the surface area of the cone

    inline double area()

    {

        //double surface\_area = M\_PI \* radius \* (radius + sqrt(pow(radius, 2) + pow(height, 2)));

        //cout << "Surface area of the cone: " << surface\_area << endl;

        //double surface\_area = M\_PI \* radius \* (radius + sqrt(pow(radius, 2) + pow(height, 2)));

       // return surface\_area;

    }

};

// initialize the static member variable

int Cone::count = 0;

int main()

{

    // create two Cone objects

    Cone cone1(2, 3);

    Cone cone2(4, 5);

    // display the number of objects created

    Cone::display(); // Output: Number of objects created: 2

    // calculate and display the surface area of the first cone

    cout<<"Surface area of cone1: "<<cone1.area(); // Output: Surface area of the cone: 37.699111843077515

    cout<<"Surface area of cone2: "<<cone2.area();

    return 0;

}

1. Write a program to overload Assignment operator (=). [Use class Result(int m1, int m2, int m3)] (15 marks)

#include <iostream>

class Result {

 public:

  // Constructor

  Result(int m1, int m2, int m3) : m1\_(m1), m2\_(m2), m3\_(m3) {}

  // Overloaded assignment operator

  Result& operator=(const Result& other) {

    m1\_ = other.m1\_;

    m2\_ = other.m2\_;

    m3\_ = other.m3\_;

    return \*this;

  }

  // Getters for the marks

  int m1() const { return m1\_; }

  int m2() const { return m2\_; }

  int m3() const { return m3\_; }

 private:

  int m1\_, m2\_, m3\_;

};

int main() {

  Result r1(50, 60, 70);

  Result r2(80, 90, 100);

  // Assign r2 to r1 using the overloaded assignment operator

  r1 = r2;

  std::cout << "r1.m1 = " << r1.m1() << std::endl;

  std::cout << "r1.m2 = " << r1.m2() << std::endl;

  std::cout << "r1.m3 = " << r1.m3() << std::endl;

  return 0;

}

1. Write a program to overload insertion and extraction operators (<<,>>). [Use class Distance(inch, feet)] (15 marks)

**#include <iostream>**

**using namespace std;**

**class Distance {**

**public:**

**int inch;**

**int feet;**

**// Constructor**

**Distance(int inch, int feet) : inch(inch), feet(feet) {}**

**Distance() : inch(0), feet(0) {}**

**// Overload the stream insertion operator (<<)**

**friend ostream& operator<<(ostream& out, const Distance& d) {**

**out << d.feet << "' " << d.inch << "\"";**

**return out;**

**}**

**// Overload the stream extraction operator (>>)**

**friend istream& operator>>(istream& in, Distance& d) {**

**in >> d.feet >> d.inch;**

**return in;**

**}**

**};**

**int main() {**

**// Create a Distance object**

**Distance d; //calls default constructor**

**// Use the overloaded stream insertion operator to print the Distance object**

**cout << d << endl;**

**cout<<"Enter feet and inches"<<endl;**

**// Use the overloaded stream extraction operator to read values for the Distance object**

**cin >> d;**

**// Use the overloaded stream insertion operator to print the modified Distance object**

**cout << d << endl;**

**return 0;**

}

1. Create a class Person(Name, Aadhar\_no). Derive classes Doctor(Speciality, place\_of\_hosp ) and Teacher(Subject, college\_name). Use appropriate constructors and destructors (15 marks)

#include <iostream>

#include <string>

using namespace std;

class Person {

 public:

  Person(string name, string aadhar\_no) : name\_(name), aadhar\_no\_(aadhar\_no) {}

  virtual ~Person() {}

  string GetName() const { return name\_; }

  string GetAadharNo() const { return aadhar\_no\_; }

 private:

  string name\_;

  string aadhar\_no\_;

};

class Doctor : public Person {

 public:

  Doctor(string name, string aadhar\_no, string speciality, string place\_of\_hosp)

      : Person(name, aadhar\_no), speciality\_(speciality), place\_of\_hosp\_(place\_of\_hosp) {}

  ~Doctor() {}

  string GetSpeciality() const { return speciality\_; }

  string GetPlaceOfHosp() const { return place\_of\_hosp\_; }

 private:

  string speciality\_;

  string place\_of\_hosp\_;

};

class Teacher : public Person {

 public:

  Teacher(string name, string aadhar\_no, string subject, string college\_name)

      : Person(name, aadhar\_no), subject\_(subject), college\_name\_(college\_name) {}

  ~Teacher() {}

  string GetSubject() const { return subject\_; }

  string GetCollegeName() const { return college\_name\_; }

 private:

  string subject\_;

  string college\_name\_;

};

int main() {

  Person p("John", "1234567890");

  cout << "Name: " << p.GetName() << ", Aadhar No: " << p.GetAadharNo() << endl;

  Doctor d("Jane", "0987654321", "Surgery", "St. Mary's Hospital");

  cout << "Name: " << d.GetName() << ", Aadhar No: " << d.GetAadharNo()

       << ", Speciality: " << d.GetSpeciality()

       << ", Place of Hospital: " << d.GetPlaceOfHosp() << endl;

  Teacher t("Mike", "1231231230", "Physics", "MIT");

  cout << "Name: " << t.GetName() << ", Aadhar No: " << t.GetAadharNo()

       << ", Subject: " << t.GetSubject() << ", College: " << t.GetCollegeName()

       << endl;

  return 0;

}

5 x 15 marks