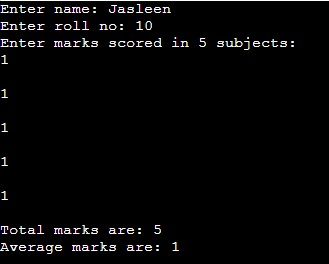
1. Consider a publishing company that markets both book and audio cassette versions to its works. Create a class Publication that stores the title (a string) and price (type float) of a publication. Derive the following two classes from the above Publication class: Book which adds a page count (int) and Tape which adds a playing time in minutes(float). Each class should have a get\_data() function to get its data from the user at the keyboard. Write the main() function to test the Book and Tape classes by creating instances of them asking the user to fill in data with get\_data() and then displaying it using put\_data()

| #include <iostream> using namespace std;  class publication{  private:  char title[50];  float price;  void getData(){  cout<<"Enter title: "<<endl;  cin>>title;  cout<<"Enter price: "<<endl;  cin>>price;  }  void putData(){  cout<<"Title is: "<<title<<endl;  cout<<"Price is: "<<price<<endl;  }  public:   void check1() { getData(); }  void check2() { putData(); } };  class book : private publication {  private:  int pageCount;  public:  void get() {  check1();  cout<<"Enter number of pages of book: "<<endl;  cin>>pageCount;  }  void put(){  check2();  cout<<"Number of pages in book are: "<<pageCount<<endl;  } };  class tape : private publication{  private:  float playingTime;  public:   void get(){  check1();  cout<<"Enter the playing time: "<<endl;  cin>>playingTime;  }  void put(){  check2();  cout<<"Playing time of tape is: "<<playingTime<<endl;  } };  int main() {  book a;  a.get();  a.put();  tape b;  b.get();  b.put();   return 0; } |
| --- |

1. Consider an example of declaring the examination result. Design three classes: student, exam and result. The student has data members such as rollno, name. Create the class exam by inheriting the student class. The exam class adds data members representing the marks scored in 5 subjects. Derive the result from exam-class and it has its own data members like total, avg. write an interactive program to model this relationship.

| #include <iostream> using namespace std;  class student {  protected:  char name[20];  int rollNo;  public:  student(){  cout<<"Enter name: ";  cin>>name;  cout<<"Enter roll no: ";  cin>>rollNo;  } };  class exam: public student{  protected:   float marks[5];  public:  exam(){  cout<<"Enter marks scored in 5 subjects: "<<endl;  for(int i=0; i<5; i++){  cin>>marks[i];  cout<<endl;  }  }  void putData(){  for(int i=0; i<5; i++)  cout<<"Marks in subject "<<i+1<<" are "<<marks[i]<<endl;  } };  class result : private exam{  private:  float tot, avg;  public:  void totMarks(){  for(int i=0; i<5; i++)  this->tot += this->marks[i];  }  void avgMarks(){  this->avg = this->tot/5;  }  void putData(){  totMarks();  avgMarks();  cout<<"Total marks are: "<<tot<<endl;  cout<<"Average marks are: "<<avg<<endl;  } };  int main() {  result res;  res.putData();   return 0; } |
| --- |



OUTPUT:

**EXTRA Question:** Multiple Base class- Constructors and Destructors in Inheritance

| #include <iostream> using namespace std;  class base1{  protected:  int i;  public:  base1(int x){  i = x;  cout<<"Constructing base1."<<endl;  }  ~base1(){  cout<<"Destructing base1."<<endl;  } };  class base2{  protected:  int k;  public:  base2(int x){  k = x;  cout<<"Constructing base2."<<endl;  }  ~base2(){  cout<<"Destructing base2."<<endl;  } };  class derived: public base1, public base2{  int j;  public:  derived(int x, int y, int z): base1(y), base2(z){  j = x;  cout<<"Constructing derived."<<endl;  }  void display(){  cout<<i<<" "<<j<<" "<<k<<endl;  }  ~derived(){  cout<<"Destructing derived."<<endl;  } };  int main() {  derived obj(2, 3, 4);  obj.display();   return 0; } |
| --- |

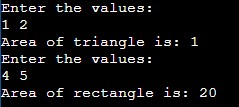
**EXTRA Question:** Virtual FUnction - Polymorphism - Function Overwritten

| #include <iostream> using namespace std;  class base{  public:  virtual void display() { cout<<"This is base class!"<<endl; } };  class derived: public base{  public:  void display() { cout<<"This is derived class!"<<endl; } };  int main() {  base B;  B.display();  derived D;  D.display(); //called due to compile time polymorphism    base \*p;  derived d;  p = &d;  p->display(); //called due to runtime polymorphism  return 0; } |
| --- |

1. (i) Create a base class called shape, Use this class to store two double type values that could be used to compute the area of figures. Derive two specific classes called triangle and rectangle from the base shape. Add tp the base class, a member function getdata() to initialize base class data members and another member function display\_area() to compute and display area of figures.

Make display\_area() as a virtual function and redefine the function in the derived class to suit their Requirements. Using these three classes, design a program that will accept dimensions of a triangle or a rectangle interactively and display area.

| #include <iostream> using namespace std;  class shape{  protected:  double a1, a2;  public:  void getData(){  cout<<"Enter the values: "<<endl;  cin>>a1>>a2;  }  virtual void displayArea(){  cout<<"Base Class"<<endl;  } };  class triangle: public shape{  private:  double area;  public:  void displayArea(){  area = 0.5 \* a1 \* a2;  cout<<"Area of triangle is: "<<area<<endl;  } };  class rectangle: public shape{  private:  double area;  public:  void displayArea(){  area = a1 \* a2;  cout<<"Area of rectangle is: "<<area<<endl;  } };  int main() {  // triangle a;  // a.getData();  // a.displayArea();  // rectangle b;  // b.getData();  // b.displayArea();    shape \*s1;  triangle objT;  s1 = &objT;  s1->getData();  s1->displayArea();    shape \*s2;  rectangle objR;  s2 = &objR;  s2->getData();  s2->displayArea();  return 0; } |
| --- |



(ii) Run the above program with following modification a)Make shape class as abstract class with display\_area() as pure virtual function b) Use constructor function to initialize the data members of base class not through the getdata().

| #include <iostream>  using namespace std;  class shape{  protected:  double a1, a2;  public:  shape(){  cout<<"Enter values: "<<endl;  cin>>a1>>a2;  }  virtual void displayArea()=0;  };  class triangle: public shape{  private:  double area;  public:  void displayArea(){  area = 0.5 \* a1 \* a2;  cout<<"Area of triangle is: "<<area<<endl;  }  };  class rectangle: public shape{  private:  double area;  public:  void displayArea(){  area = a1 \* a2;  cout<<"Area of rectangle is: "<<area<<endl;  }  };  int main()  {  // triangle t;  // t.displayArea();  // rectangle r;  // r.displayArea();  shape \*s1, \*s2;  triangle t1;  s1 = &t1;  rectangle r1;  s1->displayArea();  s2 = &r1;  s2->displayArea();  return 0;  } |
| --- |

