**Object-Oriented Programming Language**

**05/18/2023**

**Due:05/22/2023 11:59p.m.**

**Lab Assignment 11**

1. Create a class Triangle and create a class derived from the base class Triangle called Isosceles. This main code you should not change.

#include <iostream>

using namespace std;

int main(){

Isosceles isc;

isc.isosceles();

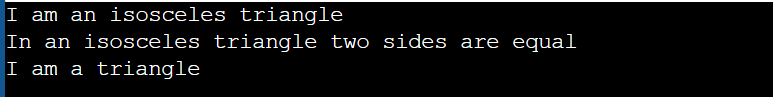
isc.description();

isc.triangle();

return 0;

}

Output should look like this



1. Design a C++ program that utilizes concepts of inheritance, protected members, and virtual functions to simulate a hierarchy of animals in a zoo.

Create a base class called Animal, which includes a protected member variable named "name" and a virtual function named "sound()".Derive two classes, Cat and Dog, from the Animal base class. Implement the sound() function in the Cat and Dog classes to display their respective sounds. Create a Zoo class, which contains a dynamic array (pointer) to store Animal objects.

This main function you should not change.

int main() {

Zoo zoo;

Cat cat("Oscar");

Dog dog("Buddy");

zoo.addAnimal(&cat);

zoo.addAnimal(&dog);

zoo.makeAllSounds();

return 0;

}

1. Write a C++ program that uses the Fraction class to represent fractions. The class should have two private member variables, numer and denom, representing the numerator and denominator of the fraction, respectively. Implement a print function that returns the string representation of the fraction. overload the input operator >> and the output operator << to read fractions from standard input and output them to standard output. This main code you should not change.

int main() {

Fraction f1(2, 4);

Fraction f2;

cin >> f2;

cout << "Fractions are: ";

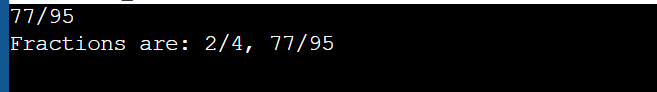
cout << f1 << ", ";

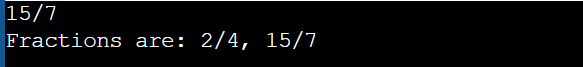
cout << f2 << endl;

return 0 ;

}

Your output should look like this.





1. Write a derived class RFraction that inherits from the Fraction class in the previous question. Override the print function in the RFraction class. The RFraction class should have an additional feature to reduce the fraction and output it. This main code you should not change.

int main() {

RFraction rf1(2, 4);

RFraction rf2;

cin >> rf2;

cout << "Reduced fractions are: ";

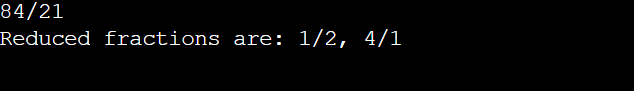
cout << rf1 << ", ";

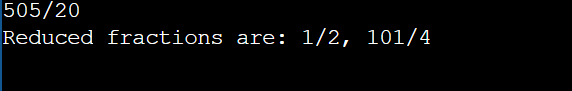
cout << rf2 << endl;

return 0;

}

Your output should look like this.





5.Write a base class Shape and derived classes Triangle, Circle, and Rectangle.

The base class Shape has two virtual functions:

double getArea();

string getClassName();

The << operator should also be overloaded for the class Shape only as the second operand.

Please complete the necessary data members, constructors, and functions so that the following main function can produce the corresponding output. This main code you should not change.

int main(){

vector<Shape\*> vs;

Rectangle r(10,20);

Circle c(10);

Triangle t(18,30,24);

vs.push\_back(&r);

vs.push\_back(&c);

vs.push\_back(&t);

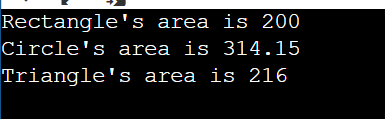
for(auto s : vs)

cout << s;

return 0;

}

Your output should look like this.



1. Write a C++ program that includes a Shark class and a derived class BabyShark. The Shark class has an age member variable and implements a constructor, a print function, and a destructor. The BabyShark class extends the Shark class and adds a babyage member variable. Implement the necessary constructor, print function, and destructor for the BabyShark class. Also, Overload the output operator << to print a Shark object or its derived classes to the standard output. This main code you should not change.

int main() {

BabyShark shark1(20);

BabyShark shark2(6);

vector<BabyShark> vec;

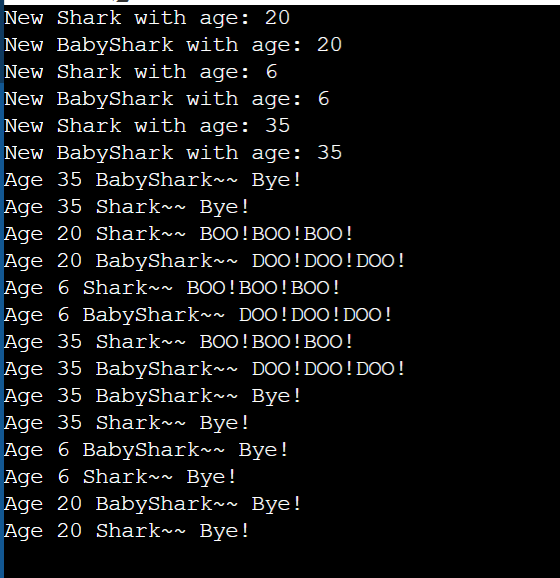
vec.push\_back(BabyShark(35));

cout << shark1 << shark2 << vec[0];

return 0;

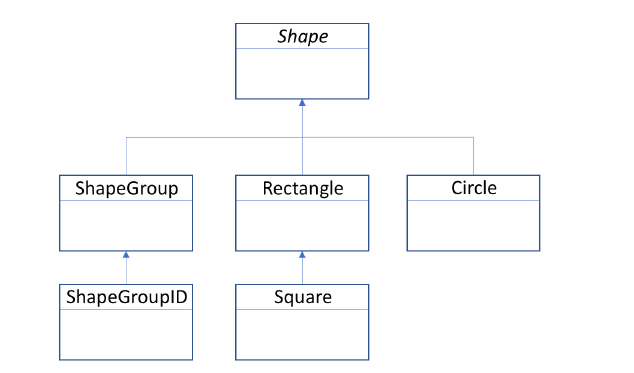
}

Your input and output should look like this.



1. Write a base class Shape and derived classes: ShapeGroup, ShapeGroupID, Triangle,Circle, and Rectangle.

The outline of the inheritance structure is as shown in the following figure:



The ShapeGroup and ShapeGroupID classes each use a dynamic array1 to store:

1. pointer to specific array of Shape\*s and

2. corresponding integer IDs for each stored Shape pointers.

To add members into the classes ShapeGroup and ShapeGroupID, the following function is used:

virtual void insert(Shape\* s);

Make sure that the parent class Shape is an Abstract Base Class.

The base class Shape has two virtual functions:

void printArea();

string getClassName();

Please complete the necessary data members, constructors (including the copy constructors), destructors and functions so that the following main function can produce the corresponding output.This main code you should not change.

int main(){

srand(time(0));

ShapeGroupID sgID;

Rectangle\* r1 = new Rectangle(10,20);

Circle\* c1 = new Circle(10);

Square\* s1 = new Square(10);

sgID.insert(r1);

sgID.insert(c1);

sgID.insert(s1);

ShapeGroupID sgID2;

Circle\* c2 = new Circle(5);

Square\* s2 = new Square(5);

sgID2.insert(c2);

sgID2.insert(s2);

ShapeGroupID sgID3(sgID);

Circle\* c3 = new Circle(20);

Square\* s3 = new Square(20);

sgID3.insert(c3);

sgID3.insert(s3);

sgID3.insert(&sgID2);

sgID3.printArea();

delete r1; delete c1; delete s1; delete c2; delete s2;

delete c3; delete s3;

return 0;

}

Your output should look like this.

