Quiz on Polymorphism

NOTE: USE THE GIVEN CODE AS REFERENCE. DO NOT JUST COPY PASTE AS THERE CAN BE SYNTAX ERRORS IN GIVEN CODE SNIPPETS.

Exercise 0 Creating Inheritance relationship:

- Create a class shape with functions and member as shown in figure
- Create classes circle square and rectangle as sub classes of class shape
- Each sub class of shape should override a calculate area of class shape according the given formula in figure.
- Your classes should have overloaded constructors that will take the member variables as input.

shape	rectangle	triangle	circle

public:
float area()
{
//definition
}
string color;
	public:
	float area()
	{
	//definition
	}
	private:
	float height;
	float width;
		public:
		float area()
		{
		//definition
		}
		private:
		float base;
		float height;
			public:
			float area()
			{
			//definition
			}
			private:
			float radius;

0.0	Area = Length X Width	Area = 1/2 of the base X the height	$A = \pi r^2$

Exercise 1 Test Case:

In your main function, create following objects

```
triangle t1(1.0,9.0, "Red")
circle c1(2, "Blue");
rectangle r1(6,2, "Orange")
cout<<t1.area();
cout<<t1.color;
cout<<c1.area();
cout<<r1.area();

shape *sptr1= &t1;
shape &sref=r1;
cout<< sptr->area()
cout<< sptr->color;
cout<< sref.color;
cout<< sref.area();
```

Q: sptr or sref invoked area of which class?

Answer:
They both called the area function of their own shape class.

Exercise 2

What if we wanted to use the area definition of the derived class function? How can you accomplish this?

Answer:
We can accomplish it by using "virtual" before area function in shape class.

```
virtual float area()
{
return 0.0;
}
```

Modify your code to use the area definition of the derived class function, compile your code, execute it and paste the output in the space provided below

Output:
4.5
Red
12.56
12
4.5
Red
Orange
12

Exercise 3a

Create a function sumArea that takes two shapes of any type and return the sum of their area (shape/rectangle/circle etc...). Note that sumArea is a nonmember function, you can declare is above your main function. What will be the type of function parameters? Prototype of function is given. Complete it and give the output.

```
int sumArea(? Shape1,? Shape2)
{
//
}

//test sumArea in main
int main
{
triangle t1(1.0,9.0, "Red")
circle c1(2, "Blue");
rectangle r1(6,2, "Orange")
shape s1("Purple");

cout<<sumArea(t1,c1);
cout<<sumArea(s1,r1);
cout<<sumArea(s1,t1);

}
```

```
Output:

17

12

4
```

Exercise 3b

Take 5 shapes as input from user. Store these in one array.
In one loop print the area of these shapes.
Use following pseudo code to take input from user (although a better way will be to overload extraction operator)
Identify the type of shape array Complete the following code and give the output.

```
int count= 5
? shapesArray=?
for(int i=0; i<count)
{
	cout << "Press 1 for a triangle, 2 for rectangle and 3 for a circle." <<
end;
	switch (getch())
	{
		case '1':
			//get base from user as input
			//get height from user as input
			//get color from user as input
			// create new triangle object and add to shapesArray[i]
			i++
			break;
		case '2':
			//get length from user as input
			//get width from user as input
			//get color from user as input
			// create new rectangle object and add to shapesArray[i]
			i++
			break;

		case '3':

			//get radius from user as input
			// create new circle object and add to shapesArray[i]
			i++
			break;

		default:
			cout<<"Invalid input. Enter again." <<endl<<endl;
			break;
}

//print area of all the shapes in shapeArray
//delete all object you have created using new.
```

Exercise 4 Virtual Destructors:

Add the following inline definitions of the destructors in their corresponding classes

```
~shape(){ cout << "~shape() called."<<endl;}
~triangle(){ cout << "~ triangle () called."<<endl;     }
~circle() { cout << "~ circle () called."<<endl; }
~rectangle(){ cout << "~rectangle() called."<<endl;     }
```

Execute the following lines in main and see output below.

```
shape *s1= new triangle(1.0,9.0, "Red");// constructor of triangle invoked
delete s1; //identify which destructor in invoked?
```

Which destructor is invoked in the last line and what is the issue in this case? How will you overcome it?

Answer:
The shape class destructor is called. Now the issue is that we want to call the destructor of the triangle class and it is not called. So it can be overcome by using "virtual" keyword before destructor of the shape class.

Exercise 5 Pure Virtual Function:

Write the following line in main and see if you can create and object of type shape
Shape S1("red");
Now make shape.area() a pure virtual function:

Code to make shape.area() a pure virtual function:

```
#include<iostream>
#include<conio.h>
using namespace std;
class shape
{
public:
	virtual float area() = 0;
	string colour;
	shape(string c)
	{
		colour = c;
	}
	~shape() {
		cout << "~shape() called." << endl;
	}
};
class rectangle :public shape
{
private:
	float height;
	float width;
public:
	float area()
	{
		float area = height * width;
		return area;
	}
	rectangle(float h, float w, string c) :shape(c)
	{
		height = h;
		width = w;
	}
	~rectangle() { cout << "~rectangle() called." << endl; }

};
class triangle :public shape
{
private:
	float base;
	float height;
```

```
public:
	float area()
	{
		float area = 0.5 * (height * base);
		return area;
	}
	triangle(float b, float h, string c) :shape(c)
	{
		base = b;
		height = h;
	}
	~triangle() { cout << "~ triangle () called." << endl; }

};
class circle :public shape
{
private:
	float radius;
public:
	float area()
	{
		float area = 3.14 * radius * radius;
		return area;
	}
	circle(float r, string c) :shape(c)
	{
		radius = r;
	}
	~circle() { cout << "~ circle () called." << endl; }
};
int sumArea(shape& Shape1, shape& Shape2)
{
	int sum = (Shape1.area() + Shape2.area());
	return sum;
}
int main()
{
	shape S1("red");

}
```

Can you create a shape type object now?

Yes or No?
No.