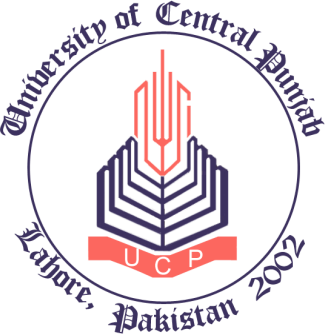
Object Oriented Programming



**Topic:**

**Polymorphism, Function Template**

Faculty of Information Technology UCP Lahore Pakistan

**Task 1**

In geometry, a shape can be defined as the form of an object or its outline, outer boundary or outer surface. Everything we see in the world around us has a shape. We can find different basic shapes such as the two-dimensional square, rectangle, and triangle. The 2-Dimensional shape is flat. It has two dimensions, that is, length and width. Every Shape has some area which can be calculated using the mathematical area function of that particular shape. The color, overall size, and orientation called the non-defining attributes of a shape do not define or affect the shape in any way. These attributes can change without any effect on the shape.

**Using the above scenario answer the following questions**

1. Identify classes attributes and their respective methods.
2. Draw the class diagram.
3. Implement the classes.
4. Make the objects of each class in main function and call all relevant function.
5. Now make the polymorphic calls for all the methods by making the appropriate objects.

**Task 2 - Function Templating**

For this part of the lab make a template out of the myMax function and test it on different data types.

* Start with the repl code provided to you.
* Compile and run the program to see how it works.
* Make a template out of myMax. Don't forget the return type.
* Modify the prototype appropriately.
* Test your myMax template on int, double, and string types.

When you are done your output should resemble this:

The max of 3 and 5 is 5

The max of 5.6 and 7.3 is 7.3

The max of donkey and apple is donkey

**Task 3:**

# Perform the following tasks.

1. An abstract class called Creature
2. Classes Player and Monster (derived from Creature)
3. Classes WildPig and Dragon (derived from Monster)

# In the Creature class

* **Define a char\* member, CreatureName,** to store the class Creature’s name.

# Two functions

**void DoAction() :** Print the action of the object, and the actions have to be different from different classes.

**void DrawOnScreen() :**Print the object’s name and call DoAction() belonging to the same class.

# The class definition of Creature is:

class Creature { public:

Creature(string);

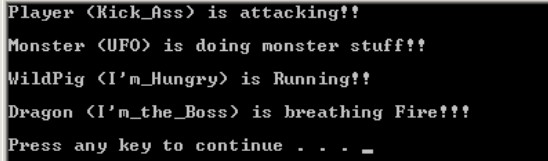
virtual void DoAction()const=0; virtual void DrawOnScreen()const=0; protected:

string CreatureName;

};

# Implement the class Player, Monster, Dragon and Wildpig so that when execution the following code, the

**counsel shows the execution result as the following: Console Output:**



**Main function:**

int main(){

Player hero("Kick\_Ass"); Monster mon("UFO"); WildPig pig("I'm\_Hungry"); Dragon drag("I'm\_the\_Boss"); Creature\* object[4]; object[0]=&hero; object[1]=&mon; object[2]=&pig; object[3]=&drag;

object[0]->DrawOnScreen(); object[1]->DrawOnScreen(); object[2]->DrawOnScreen(); object[3]->DrawOnScreen(); return 0;}