Lab Exercises

OBSERVATION QUESTIONS

1. What do you call the languages that support classes but not polymorphism?
a) Class based language
b) Procedure Oriented language
c) Object-based language
d) If classes are supported, polymorphism will always be supported

2. Which class/set of classes can illustrate polymorphism in the following code?

```
abstract class student
{
   public : int marks;
   calc_grade();}
class topper:public student
{
    public : calc_grade()
    {
        return 10;
    }};
class average:public student
{
     public : calc_grade()
     { return 20;  }
};
class failed{ int marks; };
```

a) Only class student can show polymorphism
b) Only class student and topper together can show polymorphism
c) All class student, topper and average together can show polymorphism
d) Class failed should also inherit class student for this code to work for polymorphism

3. Predict output of the following program

```
class Base
{
public:
    virtual void show() { cout<<\" In Base \\n\"; }
};

class Derived: public Base
{
public:
    void show() { cout<<\"In Derived \\n\"; }
};

int main(void)
{
    Base *bp = new Derived;
    bp->show();

    Base &br = *bp;
    br.show();

    return 0;
}
```

4. What is the advantage of declaring a virtual function as pure?

5. The derived class constructor

a) never passes any values to base class constructor
b) can pass arguments only to one base class constructor function
c) is responsible for passing the entire test of arguments needed by base class constructors
d) none of above

EXECUTION QUESTIONS

1. Implement a hierarchy of classes representing different types of vehicles. The base class Vehicle should have constructors and destructors to track the creation and destruction of vehicles. Derived classes such as Car, Truck, and Motorcycle should inherit from Vehicle and have their own constructors and destructors. Ensure that each class outputs a message when its constructor and destructor are called.

2. Implement the following scenario: an action RPG where players control different types of characters, each with unique abilities. Develop a system based on the following conditions-

1. A base class Character with a virtual function void useAbility() representing the character's ability.
2. Derived classes Wizard, Knight, and Archer that inherit from Character and override the useAbility() function to implement their specific abilities.
3. Implement constructors and destructors for each class to manage resource allocation and deallocation.
4. Create a function void playGame(Character* character) that takes a pointer to a Character object and allows the player to use the character's ability in the game (printed on screen)

3. Help implement a zoo virtually! Model these animals and their sounds by following the below given instructions-

1. A base class Animal with a virtual function void makeSound() representing the sound made by the animal.
2. Derived classes Dog, Cat, and Bird that inherit from Animal and override the makeSound() function to implement their specific sounds (e.g., barking, meowing, chirping).
3. Implement constructors for each class to initialize the animals with appropriate attributes.
4. Create a function void zooSounds(Animal** animals, int numAnimals) that takes an array of pointers to Animal objects and the number of animals, and makes each animal in the zoo make its sound.

In the program, when the zooSounds() function is called, it should iterate through the array of animals and invoke the makeSound() function for each animal, resulting in the correct sound for each animal type.