Q1. What is the meaning of multiple inheritance?
Answer: When a class inherits from more then one base class its called multiple inheritance. The child class inherits all properties of all base classes

Q2. What is the concept of delegation?
Answer: Delegation is an object oriented design pattern that allows objects to achieve the same code reuse an inheritance.

```
class Animal:
    def __init__(self, name, num_of_legs):
        self.name = name
        self.num_of_legs = num_of_legs

    def get_legs(self):
        print("the animal {0} has {1} number of legs".format(self.name,self.num_of_legs))


class Cat(Animal):
    def __init__(self,name,num_of_legs):
        super().__init__(name, num_of_legs)

cat = Cat('nutty',4)
cat.get_legs()
```

Q3. What is the concept of composition?
Answer: Composition is a concept in OOP wherein one class contains an object of one or more as an instance variable. Thereby creating objects of one class inside another, we can access the variables of one class from another. In UML a composite class has an object of a component class

Q4. What are bound methods and how do we use them?
Answer: A bound method is one that is dependent on the instance of the class as its first argument. From Python 3, all functions inside a class are by default bound methods

```
Class Animal:
        def legs(self, num):
                self.num = num
                print(“number of legs = {0}”.format(self.num))
```

Q5. What is the purpose of pseudoprivate attributes?
Answer:pseudo private attribute is a way to localize names of an attribute or a function by the class that contains it. In large class hierarchies, new functions can unknowingly overwrite functions with the same name and also internal functions can hide definitions of other functions written lower down the order. To avoid this pseudo private attributes were created.
.

```
class C1:
    def meth1(self):
        self.X=99
    def meth2(self):
        print(self.X)

class C2:
    def metha(self):
        self.X=81
    def methb(self):
        print(self.X)

class C3(C1,C2):
    pass

I = C3()
I.meth1()
I.metha()

I.meth2()
I.methb()

print(I.__dict__)
I._C1__X=99
I.__dict__
```

OUTPUT >>>

81
81   >>> the value of X is overwritten by the inherited class
{'X': 81}

{'X': 81, '_C1__X': 99}   >>> We declare a pseudo private attribute containing the private varialbe __X
In a dictionary,