L12 PROBLEM 2 (5/5 points)

Python supports a limited form of multiple inheritance, demonstrated in the following code:

```
class A(object):
    def __init__(self):
        self.a = 1
    def x(self):
        print "A.x"
    def y(self):
        print "A.y"
    def z(self):
        print "A.z"

class B(A):
    def __init__(self):
        A.__init__(self)
        self.a = 2
        self.b = 3
    def y(self):
        print "B.y"
    def z(self):
        print "B.z"

class C(object):
    def __init__(self):
        self.a = 4
        self.c = 5
    def y(self):
        print "C.y"
    def z(self):
        print "C.z"

class D(C, B):
    def __init__(self):
        C.__init__(self)
        B.__init__(self)
        self.d = 6
    def z(self):
        print "D.z"
```

Which `__init__` methods are invoked and in which order is determined by the coding of the individual `__init__` methods.

When resolving a reference to an attribute of an object that's an instance of class `D`, Python first searches the object's instance variables then uses a simple left-to-right, depth first search through the class hierarchy. In this case that would mean searching the class `C`, followed the class `B` and its superclasses (ie, class `A`, and then any superclasses it may have, et cetera).

With the definitions above if we define

```
obj = D()
```

then what is printed by each of the following statements?

1. `print obj.a`

   2 **Answer:** 2

2. `print obj.b`

3

Answer: 3

3. `print obj.c`

5

Answer: 5

4. `print obj.d`

6

Answer: 6

5. `obj.x()`

A.x

Answer: A.x

6. `obj.y()`

C.y

Answer: C.y

7. `obj.z()`

D.z

Answer: D.z

Check | Hide Answer

Help

Show Discussion

New Post