Lecture 9 Worksheet: OOP and Recursion

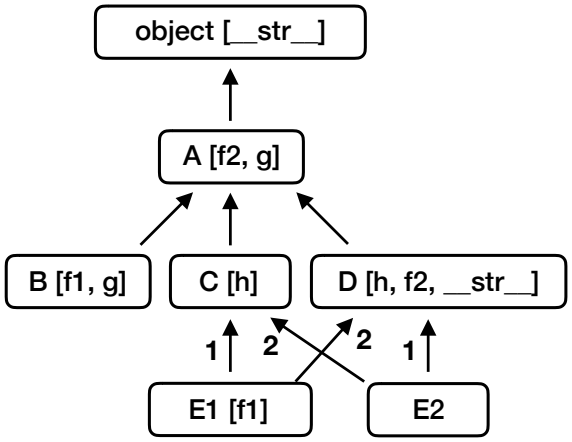
1

1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?
2. how many arguments does line C pass?
3. how many arguments does line B pass?
4. on another paper, draw what the frames and object(s) will look like after line A. (check with PythonTutor)

```
class Pet:
  def __init__(self, name):
    self.name = name # A

class Dog(Pet):
  def __init__(self, name, age):
    self.age = age
    Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

2

```
class A:
    # methods: f2, g
class B(A):
    # methods: f1, g
class C(A):
    # methods: h
class D(A):
    # methods: h, f2, __str__
class E1(C, D):
    # methods: f1
class M(B, E1):
    # methods: none



w = C()
x = M()
y = E1()
z = E2()
```



1. add code for class `E2` above
2. draw any missing details in the class hierarchy
3. what method will `x.g()` invoke?
4. what method will `x.f1()` invoke?
5. what method will `x.f2()` invoke?
6. what method will `print(x)` invoke?
7. what method will `print(w)` invoke?
8. what method will `y.h()` invoke?
9. what method will `z.h()` invoke?
10. which is correct?
    `w.__mro__` or `C.__mro__`

11. Circle Everything True

`type(w) == A`	`type(w) == C`	`isinstance(z, A)`	`isinstance(B, A)`
`type(y) == M`	`type(y) == E1`	`isinstance(w, A)`	`isinstance(w, C)`
		`isinstance(y, M)`	`isinstance(y, E1)`

3

```
def fact(n):
    if n == 0:
        return 1
    return n * fact(n-1)

# what is fact(5)
```

```
def fib(n):
    if n < 2:
        return n
    return fib(n-1) + fib(n-2)

# what is fib(6)?
```

4

```
def f(n):
    print(n)
    if n < 9:
        f(n + 1)

# what does f(7) print?
```

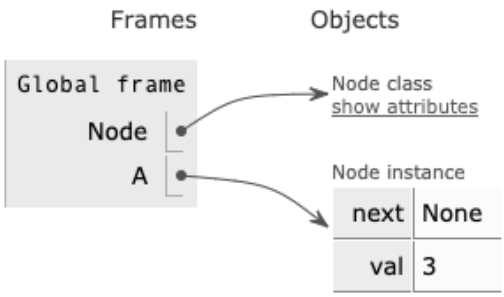
```
def g(n):
    if n < 9:
        g(n + 1)
    print(n)

# what does f(7) print?
```

```
def M(n):
    print(n)
    if n > 1:
        M(n-1)
        print(n)

# what does M(3) print?
```

```
B = []
def h(A):
    if len(A) > 0:
        h(A[1:])
        B.append(A[0])
h([2, 5, 6, 3])
# what is in B?
```

5

```
class Node:
   def __init__(self, val):
      self.val = val
      self.next = None

   def tot(self):
      if self.next != None:
         return self.val
      return self.val + self.next.tot()

   def __getitem__(self, idx):
      if idx == 0:
         return self.val
      return self.next[idx-1]

A = Node(3)
B = Node(5)
C = Node(7)
A.next = B
B.next = C
```



1. finish the PythonTutor picture on the right
2. what is `C.tot()`? `B.tot()`? `A.tot()`?
3. what is `A[0]`? `A[2]`?
4. what kind of error does `A[-1]` produce?
5. how would the PythonTutor change if we added `C.next = A`?
6. what would `C[3]` be, given above change?
7. what would `A.tot()` do, give above change?