[320] OOP and Recursion

Department of Computer Sciences
University of Wisconsin-Madison

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
```

1

1.the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

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
```

1

1.the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2.how many arguments does line C pass?

3

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
```

1

1.the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2.how many arguments does line C pass?

3

3.how many arguments does line B pass?

2

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

```
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)
```

2

```
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)
```

fact(5)

2

```
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)
```

fact(5) = 5 * fact(5)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2)

fact(2) = 2 * fact(1)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2)

fact(2) = 2 * fact(1)

fact(1) = 1* fact(0)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2)

fact(2) = 2 * fact(1)

fact(1) = 1* fact(0)

fact(0) = 1

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2)

fact(2) = 2 * fact(1)

fact(1) = 1* fact(0)

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2)

fact(2) = 2 * fact(1)

fact(1) = 1* fact(0) = 1 * 1

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2)

fact(2) = 2 * fact(1)

fact(1) = 1* fact(0) = 1 * 1 = 1

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2)

fact(2) = 2 * fact(1) = 2 * 1

fact(1) = 1* fact(0) = 1 * 1 = 1

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2)

fact(2) = 2 * fact(1) = 2 * 1 = 2

fact(1) = 1* fact(0) = 1 * 1 = 1

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2) = 2 * 2

fact(2) = 2 * fact(1) = 2 * 1 = 2

fact(1) = 1* fact(0) = 1 * 1 = 1

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3)

fact(3) = 3 * fact(2) = 2 * 2 = 6

fact(2) = 2 * fact(1) = 2 * 1 = 2

fact(1) = 1* fact(0) = 1 * 1 = 1

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3) = 4 * 6

fact(3) = 3 * fact(2) = 2 * 2 = 6

fact(2) = 2 * fact(1) = 2 * 1 = 2

fact(1) = 1* fact(0) = 1 * 1 = 1

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5)

fact(4) = 4 * fact(3) = 4 * 6 = 24

fact(3) = 3 * fact(2) = 2 * 2 = 6

fact(2) = 2 * fact(1) = 2 * 1 = 2

fact(1) = 1* fact(0) = 1 * 1 = 1

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5) = 5 * 24

fact(4) = 4 * fact(3) = 4 * 6 = 24

fact(3) = 3 * fact(2) = 2 * 2 = 6

fact(2) = 2 * fact(1) = 2 * 1 = 2

fact(1) = 1* fact(0) = 1 * 1 = 1

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5) = 5 * 24 = 120

fact(4) = 4 * fact(3) = 4 * 6 = 24

fact(3) = 3 * fact(2) = 2 * 2 = 6

fact(2) = 2 * fact(1) = 2 * 1 = 2

fact(1) = 1* fact(0) = 1 * 1 = 1

fact(0) = 1 (base case)

2

```
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)
```

fact(5) = 5 * fact(5) = 5 * 24 = 120

fact(4) = 4 * fact(3) = 4 * 6 = 24

fact(3) = 3 * fact(2) = 2 * 2 = 6

fact(2) = 2 * fact(1) = 2 * 1 = 2

fact(1) = 1* fact(0) = 1 * 1 = 1

fact(0) = 1 (base case)

Therefore, fact(5) = 120

3

```
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 g(7)
print?
```

3

```
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 g(7)
print?
```

f(7)

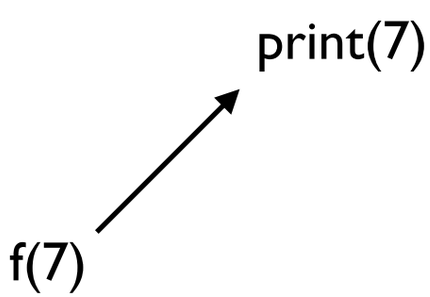
3

```
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 g(7)
print?
```



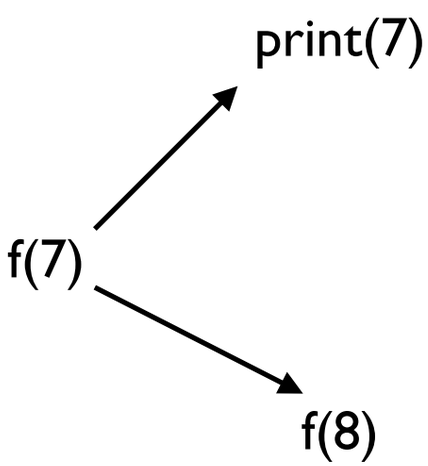
3

```
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 g(7)
print?
```



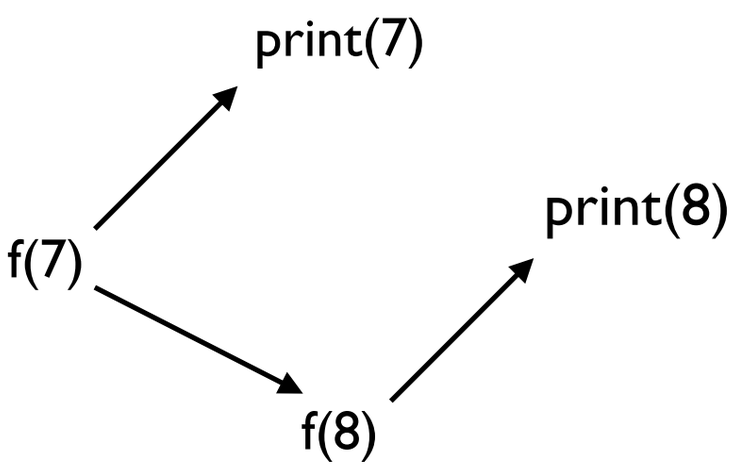
3

```
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 g(7)
print?
```



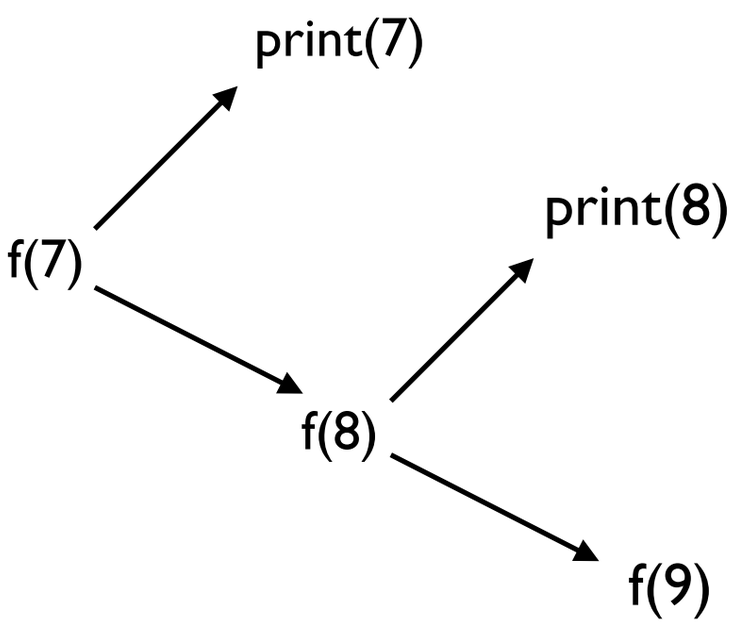
3

```
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 g(7)
print?
```



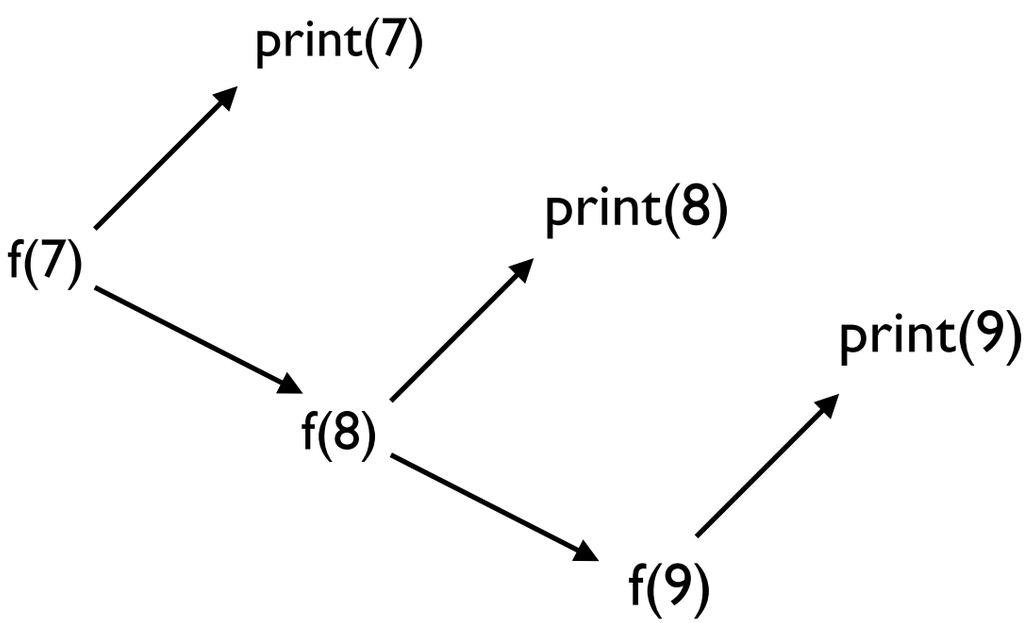
3

```
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 g(7)
print?
```



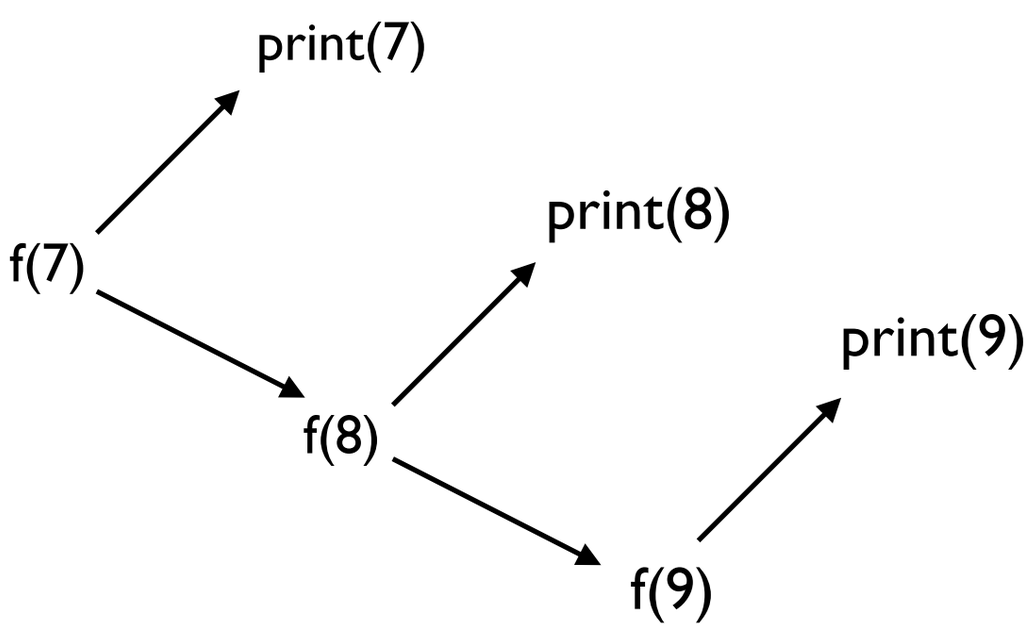
3

```
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 g(7)
print?
```



Timeline

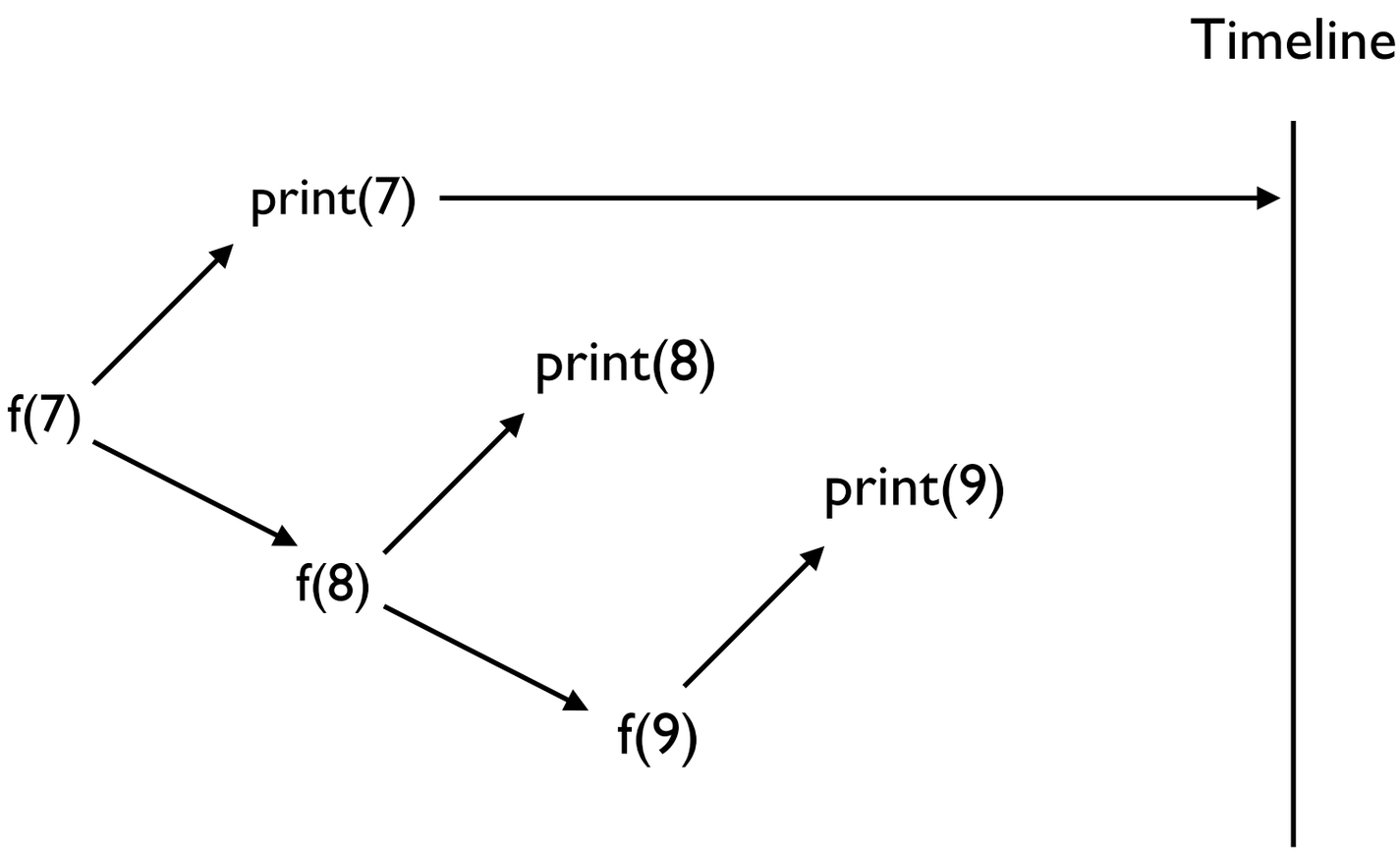
3

```
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 g(7)
print?
```



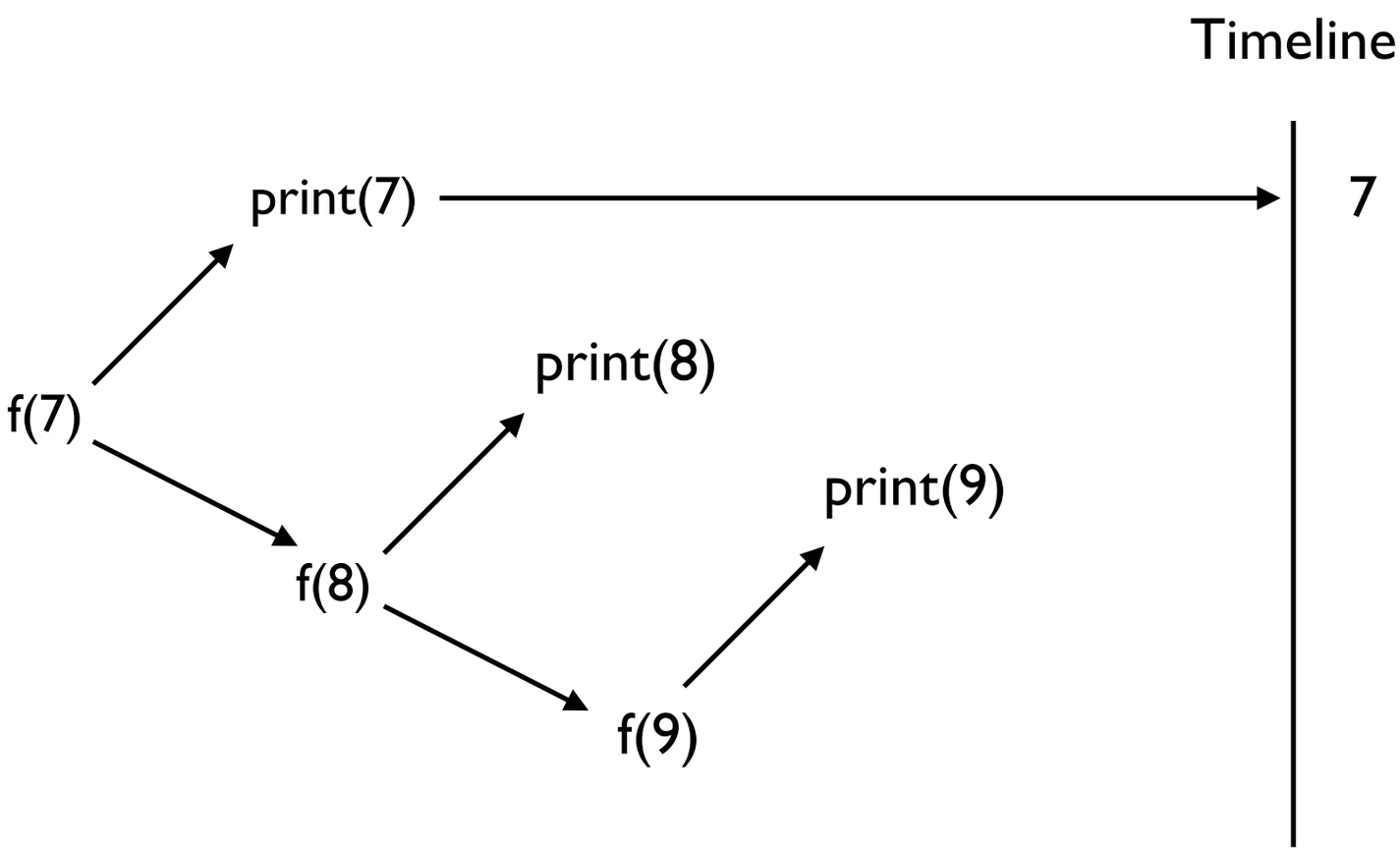
3

```
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 g(7)
print?
```



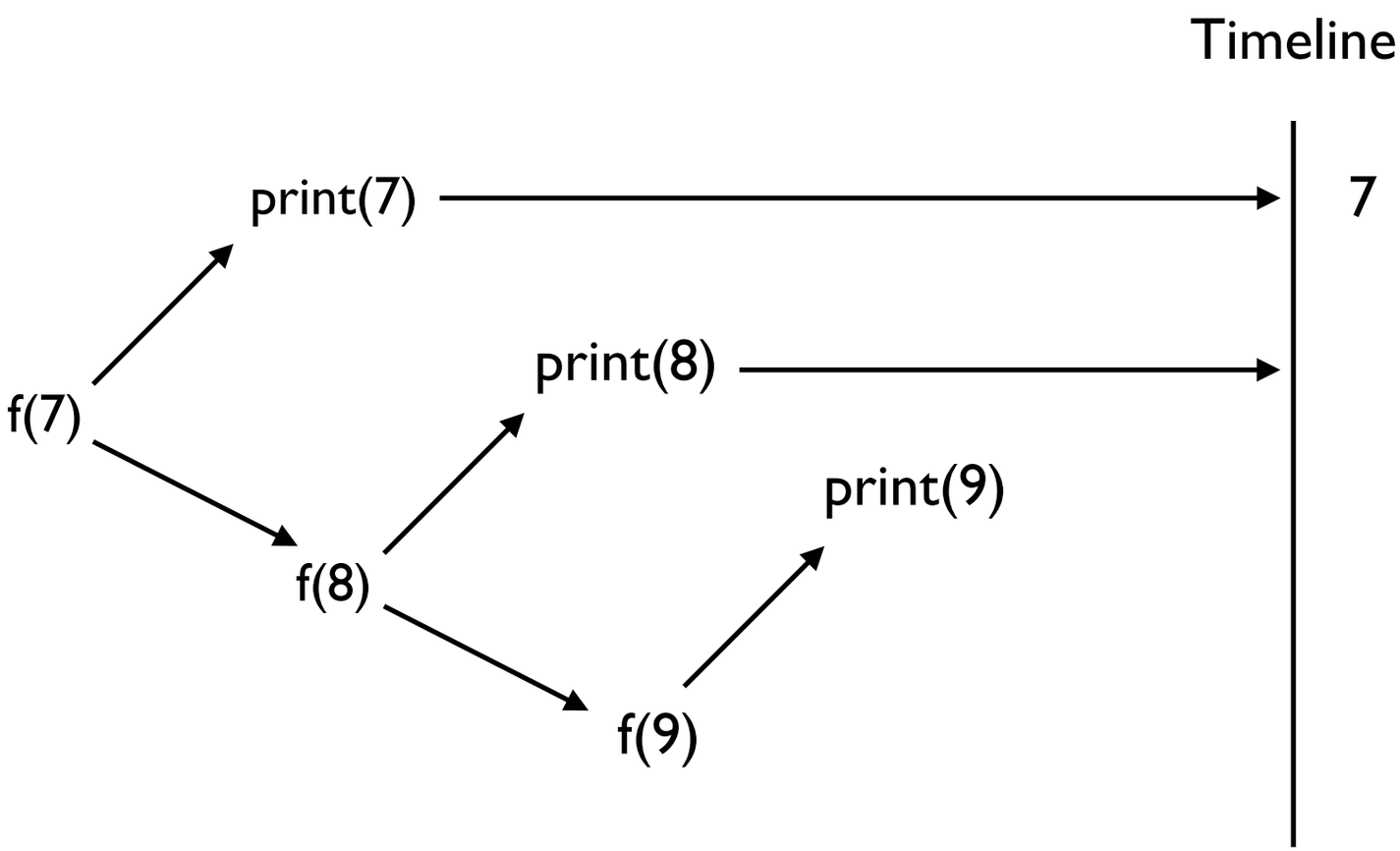
3

```
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 g(7)
print?
```



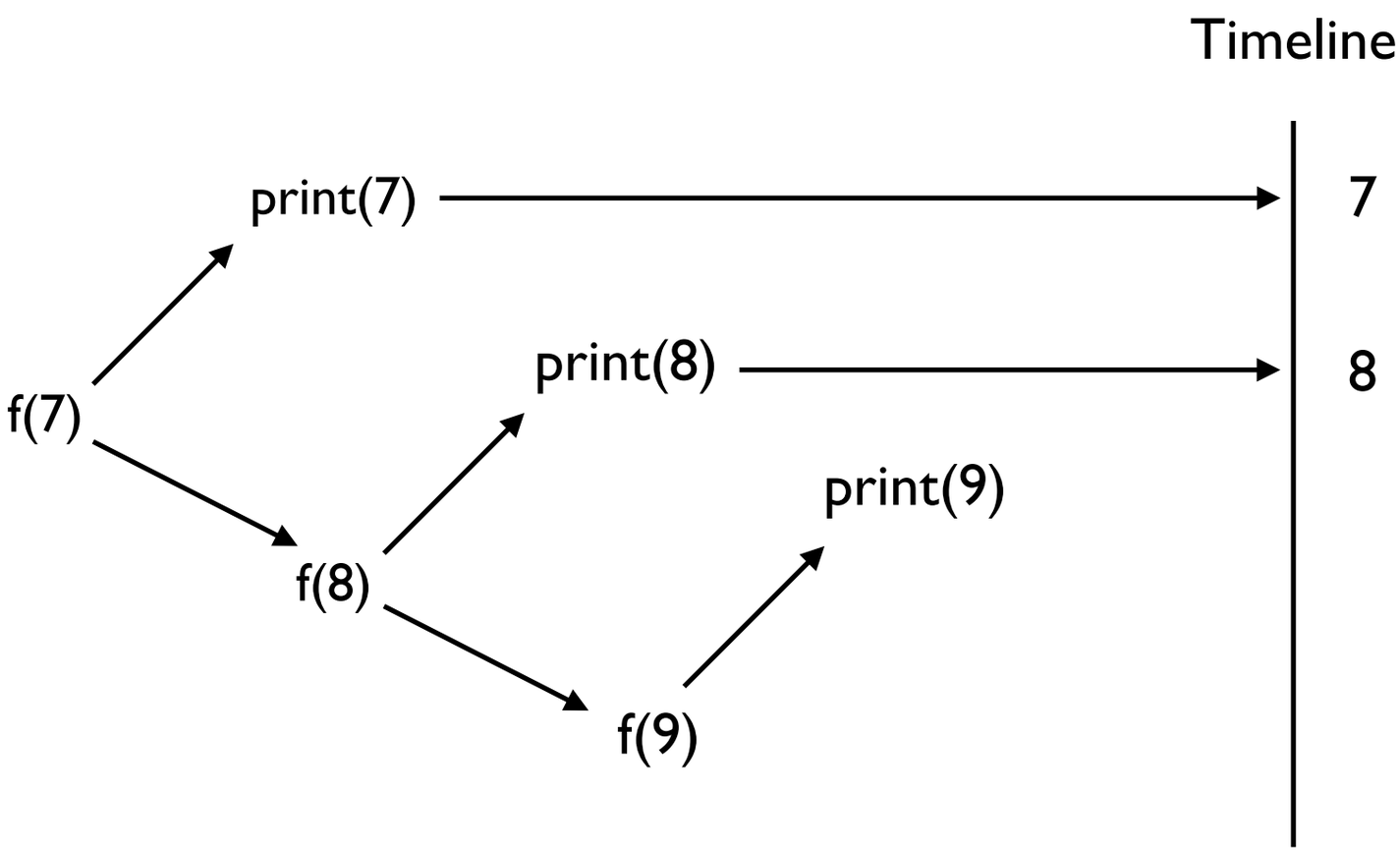
3

```
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 g(7)
print?
```



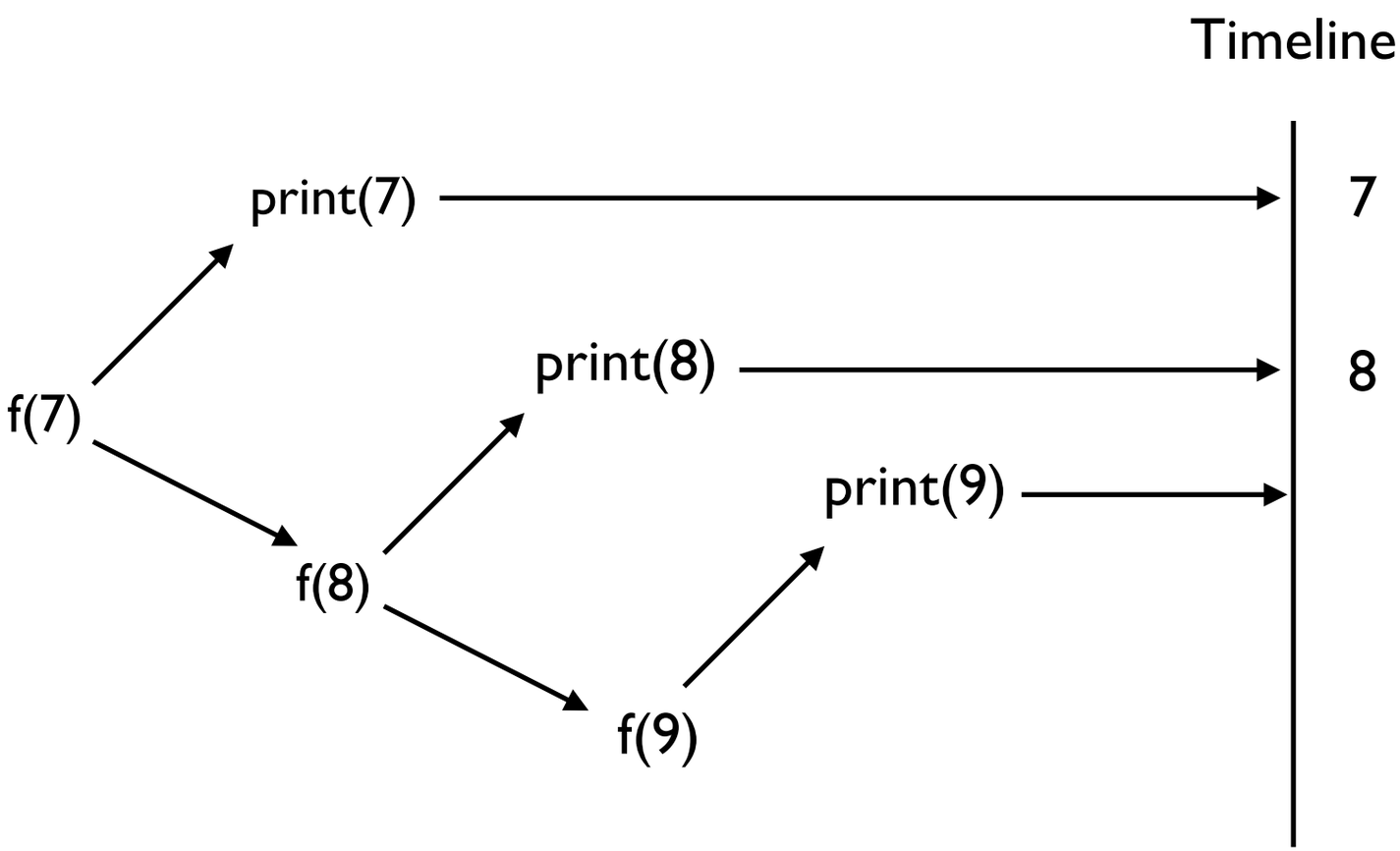
3

```
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 g(7)
print?
```



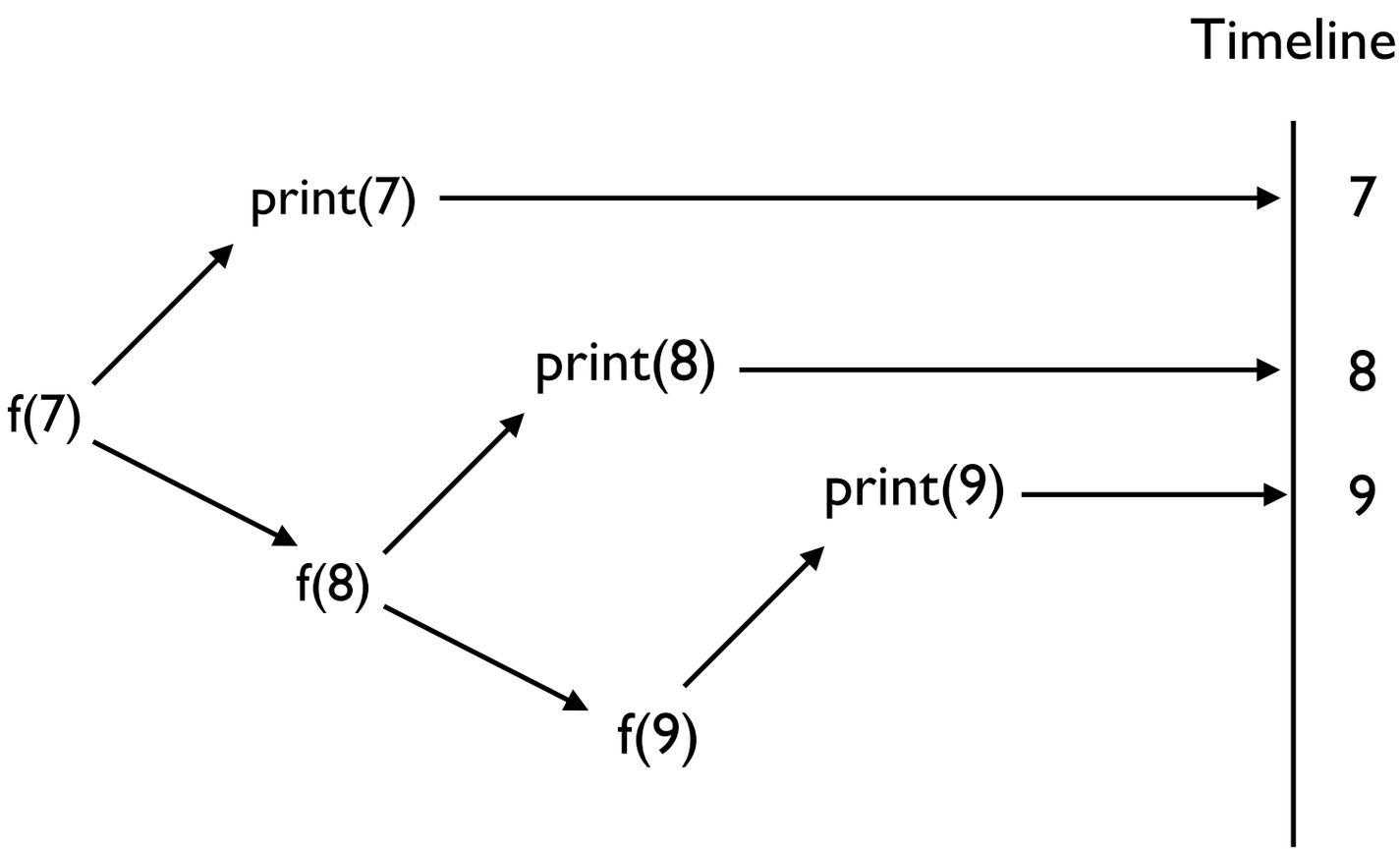
3

```
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 g(7)
print?
```



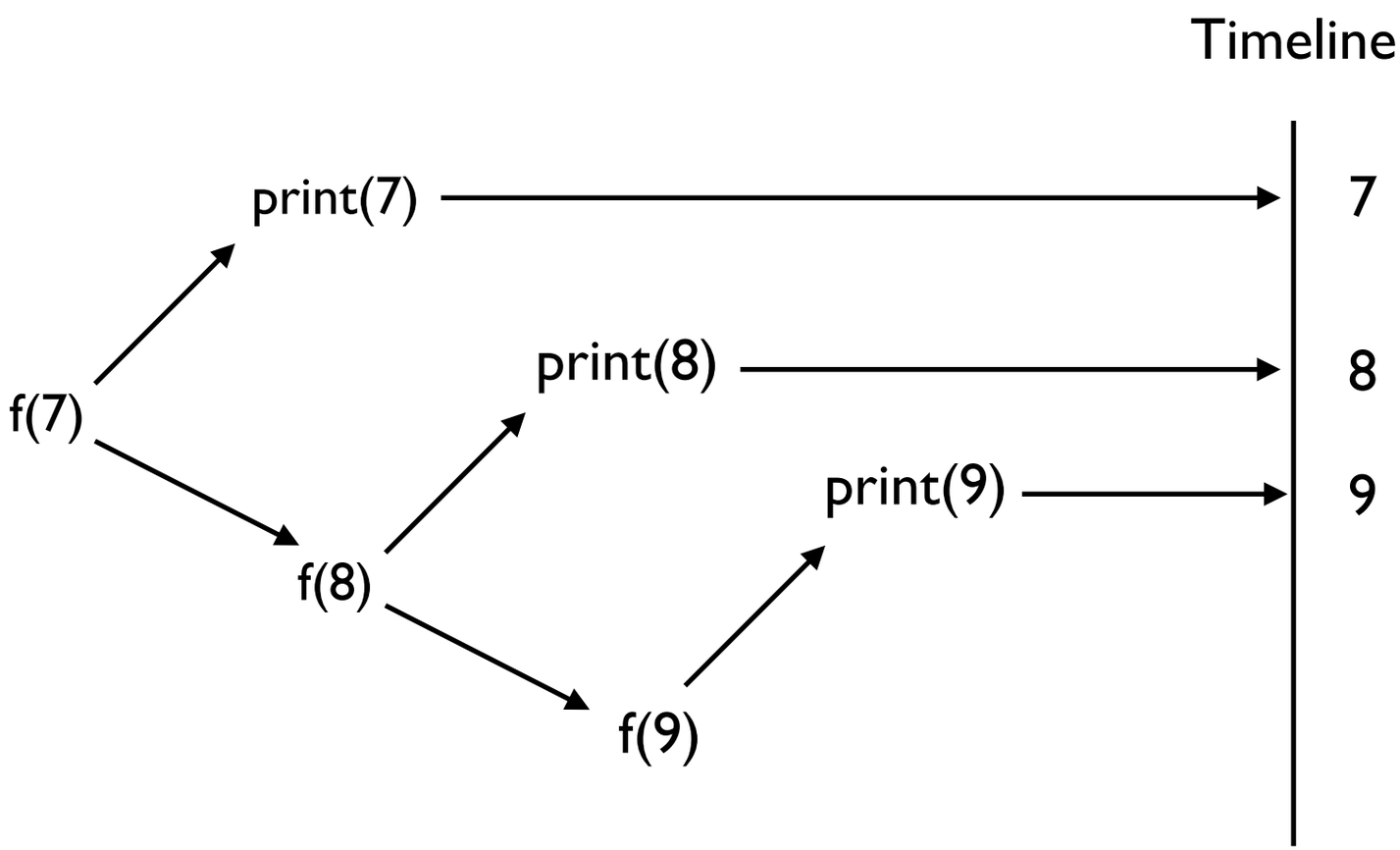
3

```
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 g(7)
print?
```



Answer: 7, 8, 9

3

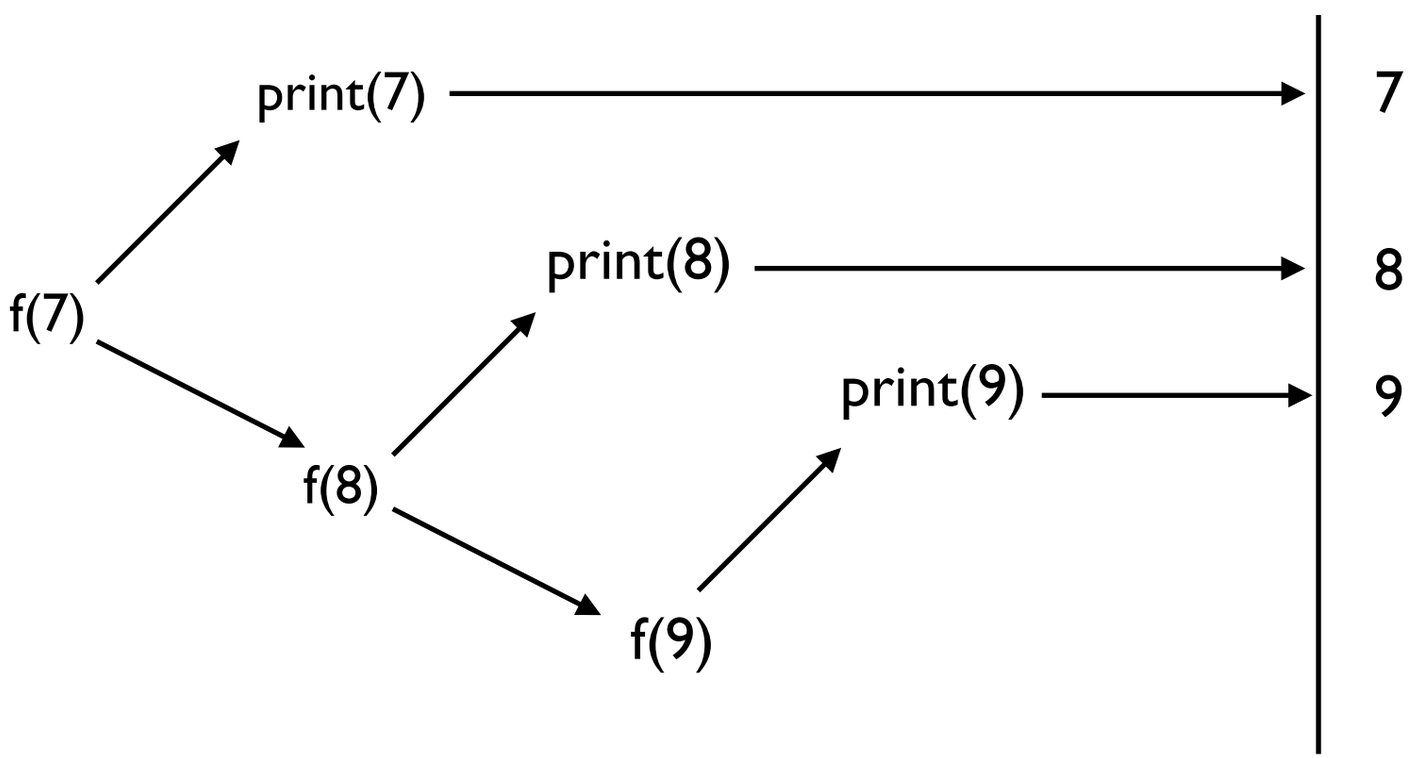
```
def f(n):
    print(n)
    if n < 9:
        f(n + 1)

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

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

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

Call graph



Timeline

Answer: 7, 8, 9

4

```
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?
```

```
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?
```

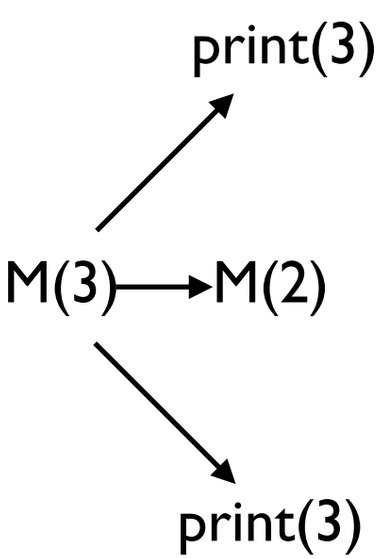
M(3)

```
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?
```

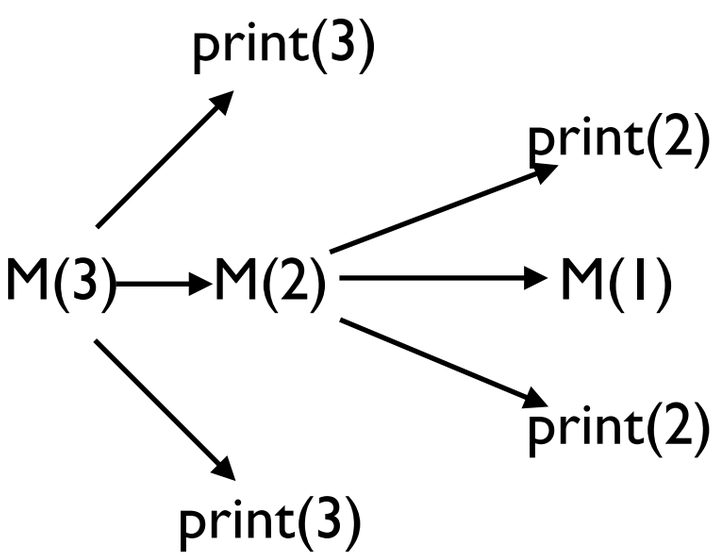


```
def M(n):
    print(n)
    if n > 1:
        M(n-1)
        print(n)

# 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?
```

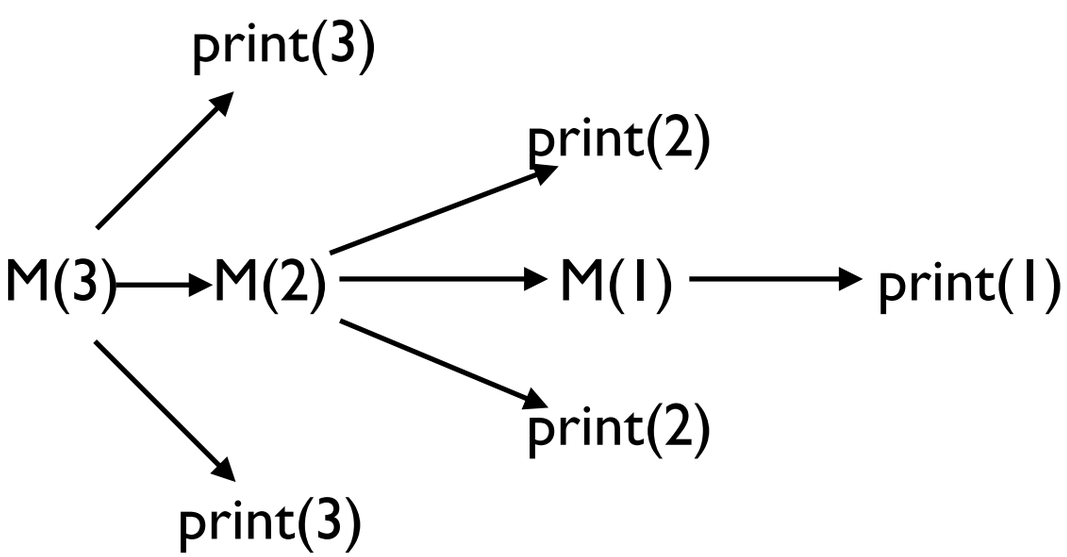


```
def M(n):
    print(n)
    if n > 1:
        M(n-1)
        print(n)

# 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?
```



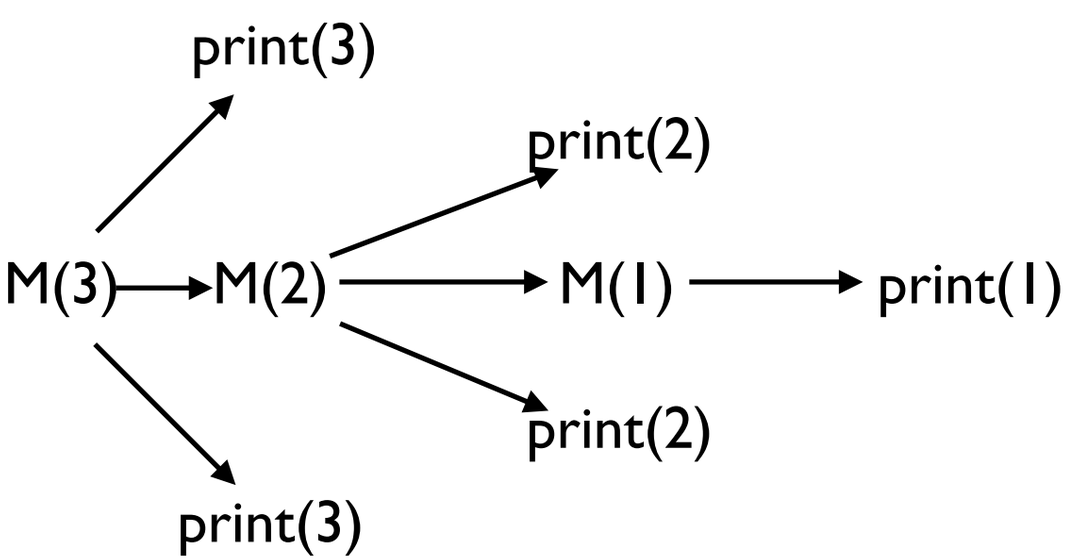
```
def M(n):
    print(n)
    if n > 1:
        M(n-1)
        print(n)

# 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?
```

Timeline

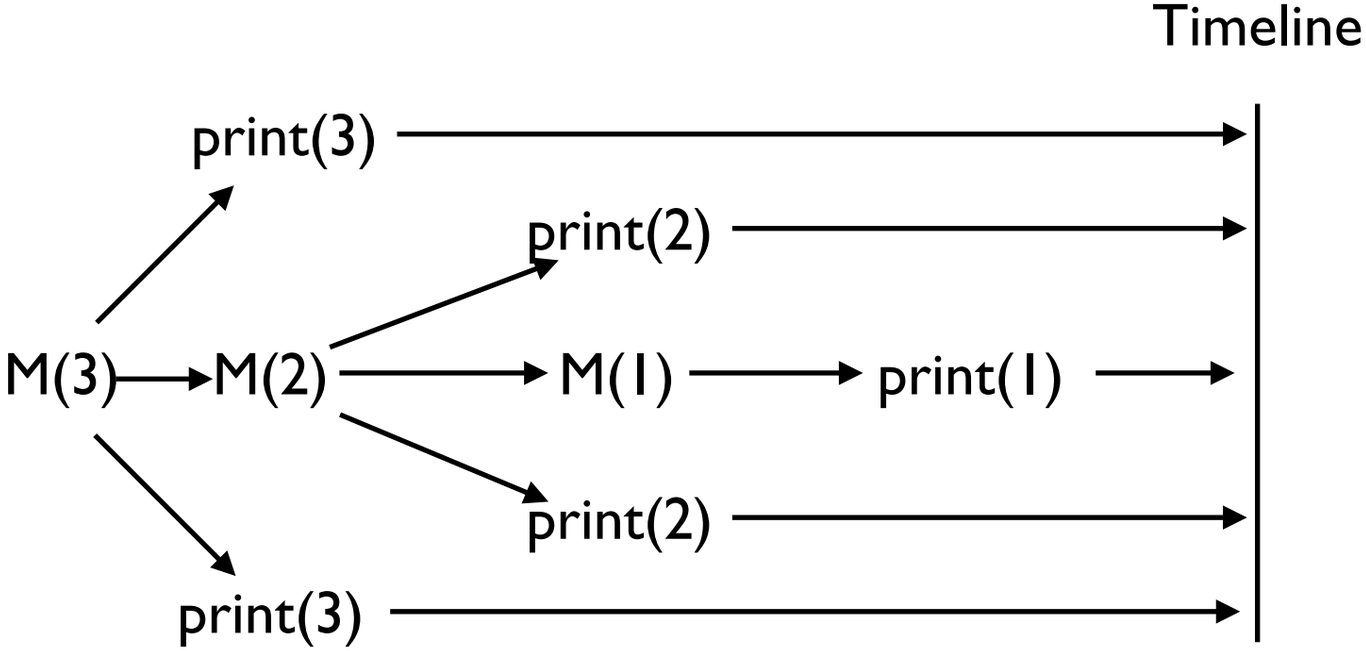


```
def M(n):
    print(n)
    if n > 1:
        M(n-1)
        print(n)

# 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?
```

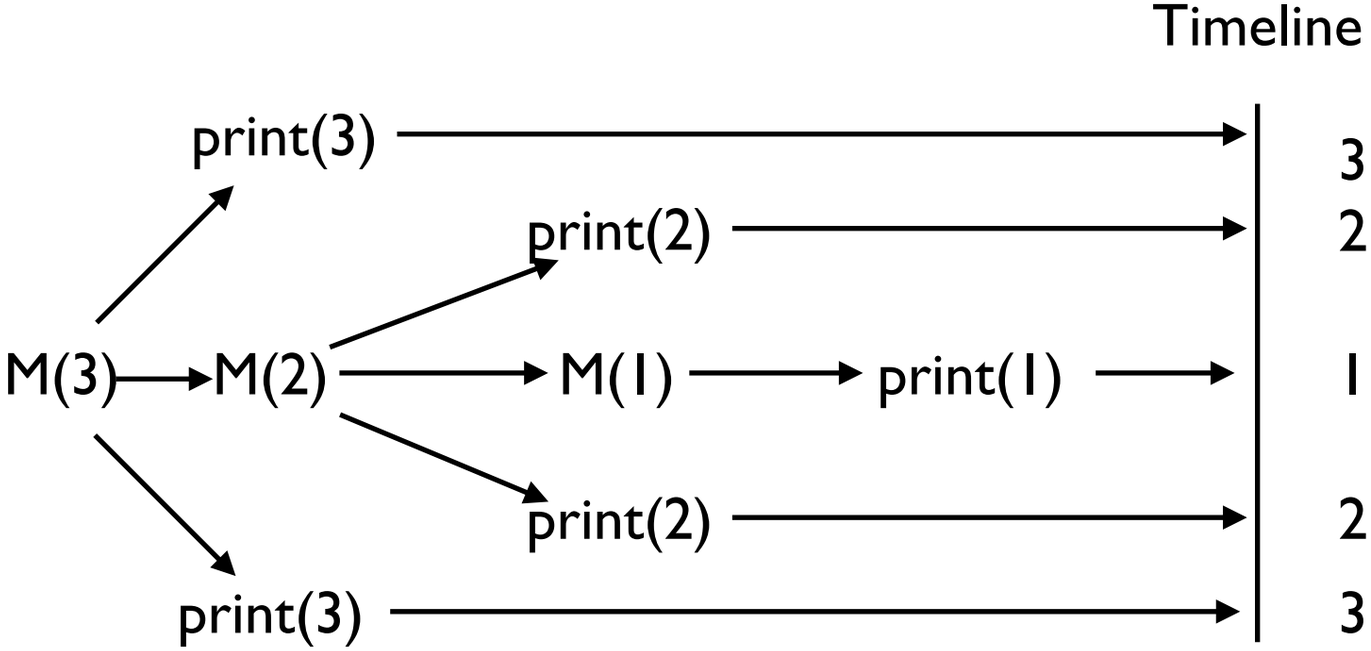


```
def M(n):
    print(n)
    if n > 1:
        M(n-1)
        print(n)

# 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?
```

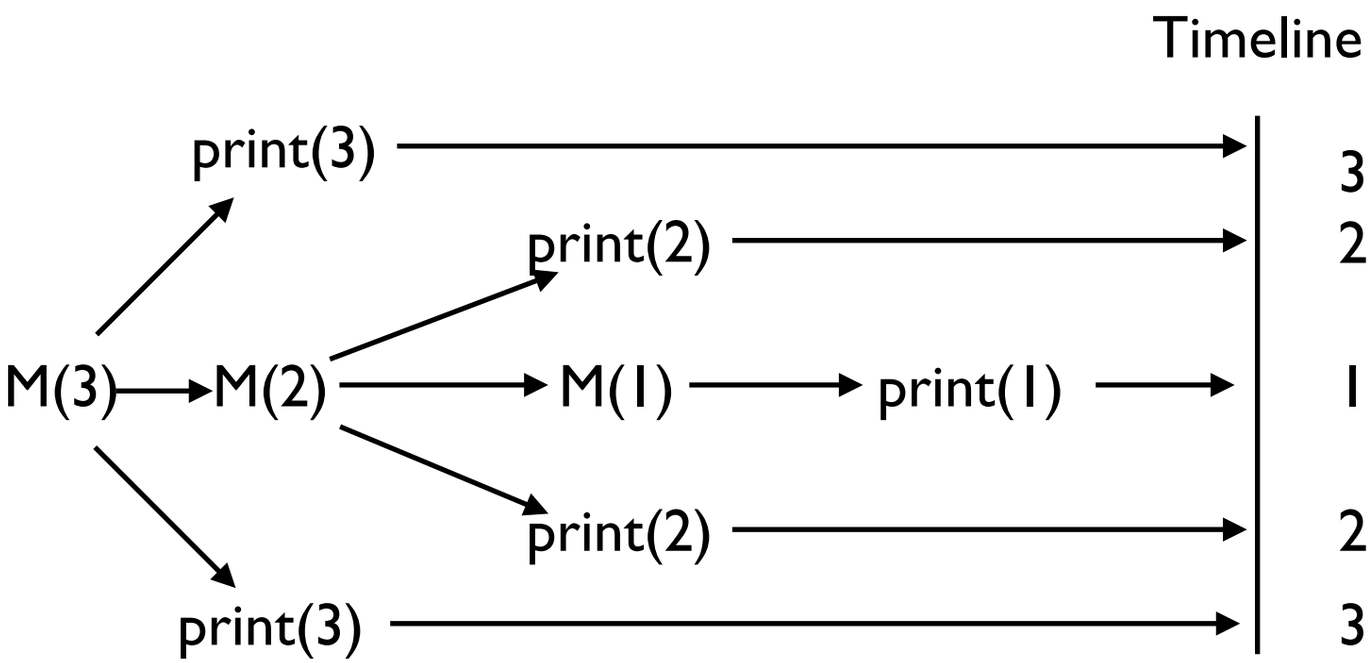


```
def M(n):
    print(n)
    if n > 1:
        M(n-1)
        print(n)

# 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?
```



Answer: 3, 2, 1, 2, 3

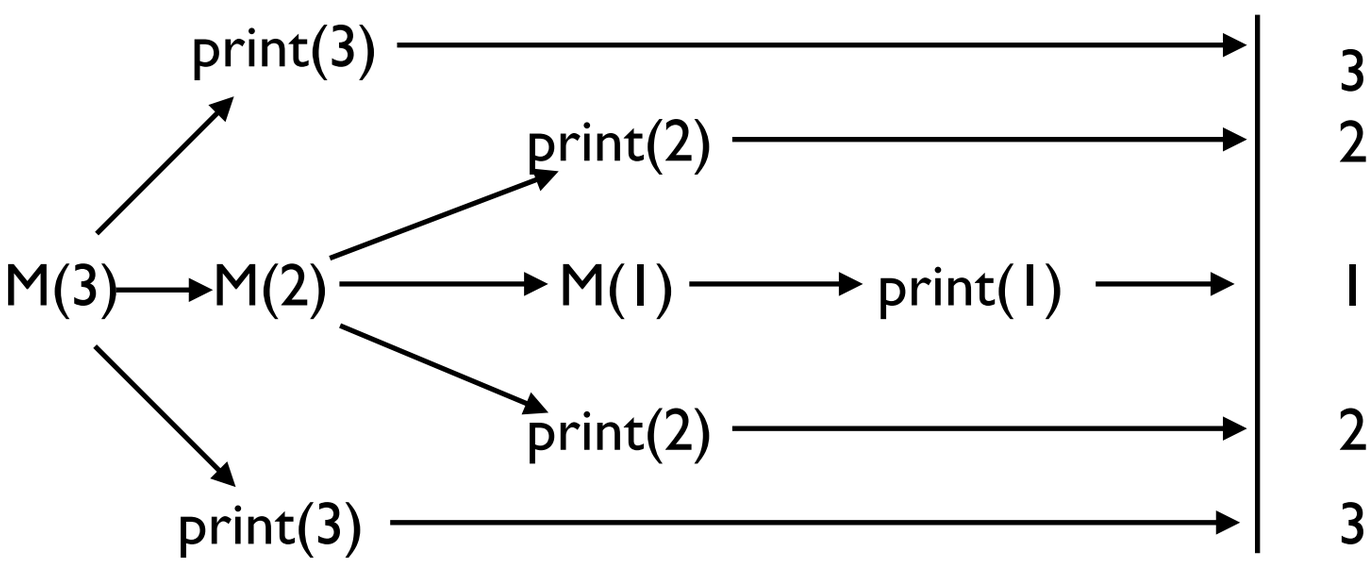
```
def M(n):
    print(n)
    if n > 1:
        M(n-1)
        print(n)

# 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?
```

Call graph



Timeline

Answer: 3, 2, 1, 2, 3