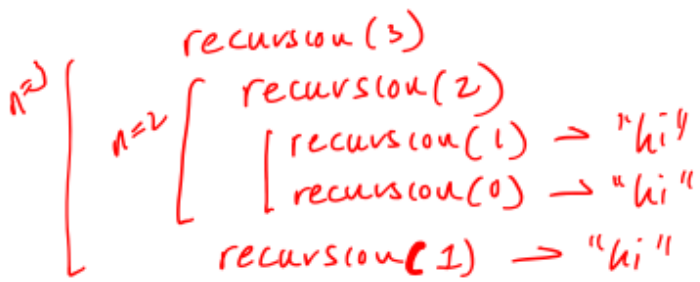
```
def recursion(n):
    if n <= 1:
        print("hi")
        return
    recursion(n-1)
    recursion(n-2)
```



How many times will “hi” be printed when you invoke `recursion(3)`?

A. 0
B. 1
C. 2
D. 3
E. 4

Answer: D
The call stack will look like:
recursion(3)
    recursion(2)
        recursion(1) -> “hi”
        recursion(0) -> “hi”
    recursion(1) -> “hi”
So “hi” should be printed 3 times.

```
def recursion(n):
    if n <= 2:
        print("hi")
        return
    recursion(n-1)
    recursion(n-2)
```

recursion(3)
recursion(2) → "hi"
recursion(1) → "hi"

How many times will “hi” be printed when you invoke `recursion(3)`?

A. 0
B. 1
C. 2
D. 3
E. 4

Answer: C
The call stack will look like:
recursion(3)
    recursion(2) -> “hi”
    recursion(1) -> “hi”
So “hi” should be printed 2 times.

1.

```
def mystery(n):
 if n <= 1:
  return 1
 else:
  return (n-1)*mystery(n)
```

2.

```
def mystery(n):
 if n <= 1:
  return 1
 else:
  return n*mystery(n-1)
```

3.

```
def mystery(seq):
 if len(seq) == 0:
  return 0
 else:
  return 1+mystery(seq[:len(seq)])
```

"hello"[:5] → "hello"
01234

Which of the code snippets above will recurse infinitely?

A. 1 only
B. 3 only
C. 1,2
D. 1,3
E. All

Answer: D

In both 1 and 3 the recursive case does not get any smaller. In the latter, we are just copying the list (we would want `seq[:len(seq)-1]`).

1.

```
def mystery(n):
 if n == 0:
  return
 else:
  mystery(n-1)
  print("Unwinding:", n)
```

2.

```
def mystery(n):
 if n <= 1:
  return 1
 else:
  return n * mystery(n-1)
```

3.

```
def mystery(n, acc):
 if n == 0:
  return acc
 else:
  return mystery(n-1, acc*n)
```

Which of the code snippets above have pending operations?

A. 1 only
B. 1,2
C. 1,3
D. 1,3
E. All

Answer: B

Both 1 and 2 have operations performed after the recursive call. For 1 it is the print, for 2 it is the multiplication by n. While for 3 no operations are performed after the recursive call, we just return the recursive result immediately.