Recursive method tracing Stack based approach Simple Example

By: Brandon Horn

apcstutoring.com

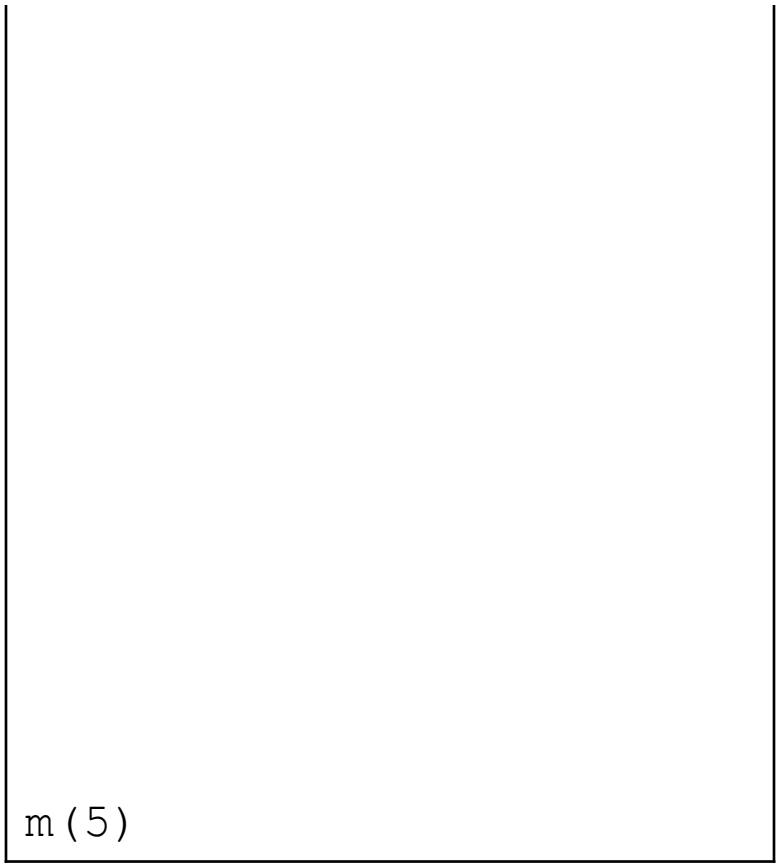
What does `mystery(5)` return?

```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3;
  else
    return mystery(b - 1) + 2;
}
```

There are 2 recursive calls in `mystery`. Label the recursive calls 1 & 2.

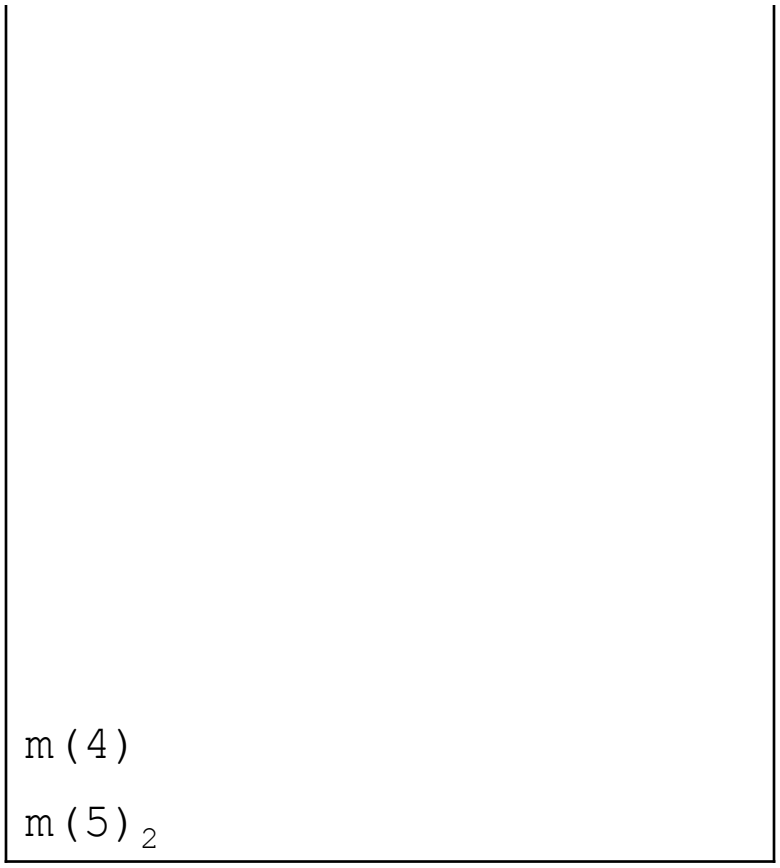
```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

Use a stack to keep track of the method calls and the return values.
The initial call is `mystery(5)`. Abbreviate the method name as `m`.



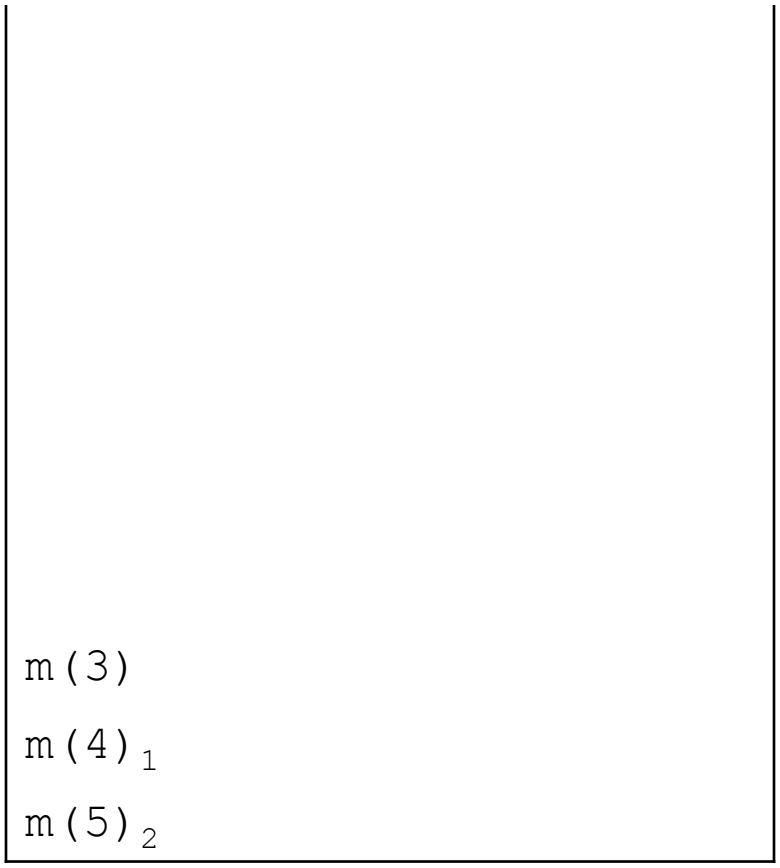

```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

`m(5)` stops at the line labeled Call 2 when it calls `m(4)`.
Use a subscript to note the call `m(5)` stopped at.
Add the new call to `m(4)` to the top of the stack.




```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

`m(4)` stops at Call 1 and calls `m(3)`.




```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

m(2)

m(3) 2

m(4) 1

m(5) 2

```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

m(1)

m(2) 1

m(3) 2

m(4) 1

m(5) 2

```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

m(0)

m(1) 2

m(2) 1

m(3) 2

m(4) 1

m(5) 2

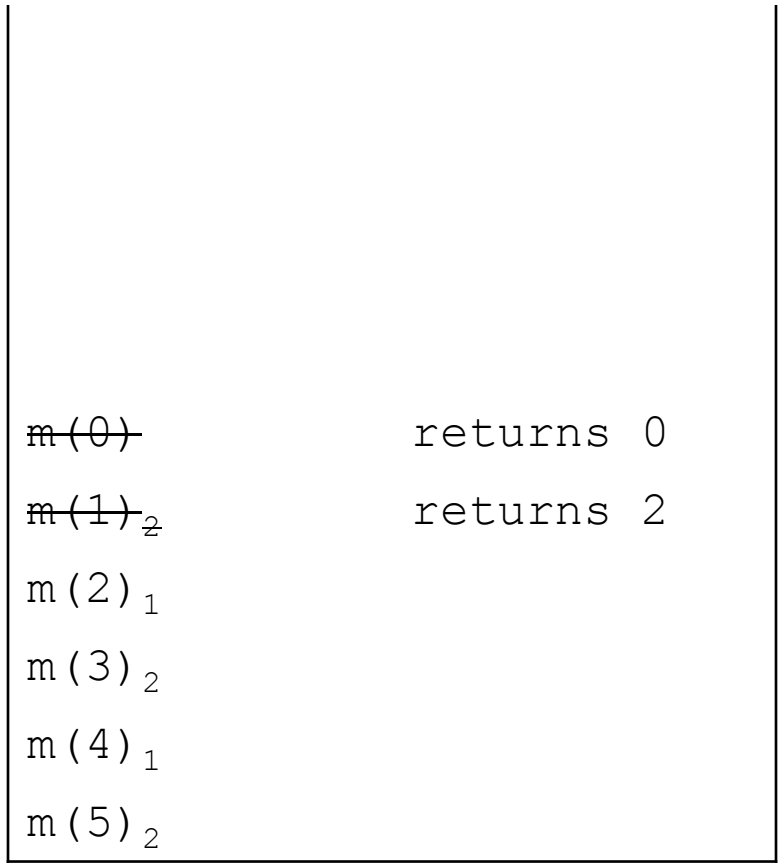
```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

`m(0)` returns 0 and terminates.
Cross out the call to `m(0)` to indicate that the method has terminated.
Write the return value to the right of the method.

~~`m(0)`~~	`returns 0`
`m(1)` 2	
`m(2)` 1	
`m(3)` 2	
`m(4)` 1	
`m(5)` 2	

```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

Control returns to the topmost non-terminated method on the stack, `m(1)`.
`m(1)` was suspended on the line labeled Call 2 when it called `m(0)`.
The call to `m(0)` returned 0.
`m(1)` returns 0 + 2 = 2.




```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

Control returns to `m(2)` which was suspended on the line labeled Call 1.
The call to `m(1)` returned 2.
`m(2)` returns 2 + 3 = 5.

~~m(0)~~	returns 0
~~m(1)~~ 2	returns 2
~~m(2)~~ 1	returns 5
m(3) 2	
m(4) 1	
m(5) 2	

```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

~~m(0)~~ returns 0

~~m(1)~~ ~~2~~ returns 2

~~m(2)~~ ~~1~~ returns 5

~~m(3)~~ ~~2~~ returns 7

m(4) 1

m(5) 2

```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

~~m(0)~~ returns 0

~~m(1)~~ $_{2}$ returns 2

~~m(2)~~ $_{1}$ returns 5

~~m(3)~~ $_{2}$ returns 7

~~m(4)~~ $_{1}$ returns 10

m(5) $_{2}$

```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```

~~m(0)~~	returns 0
~~m(1)~~ 2	returns 2
~~m(2)~~ 1	returns 5
~~m(3)~~ 2	returns 7
~~m(4)~~ 1	returns 10
~~m(5)~~ 2	returns 12

```
public int mystery(int b)
{
  if (b == 0)
    return 0;

  if (b % 2 == 0)
    return mystery(b - 1) + 3; // Call 1
  else
    return mystery(b - 1) + 2; // Call 2
}
```