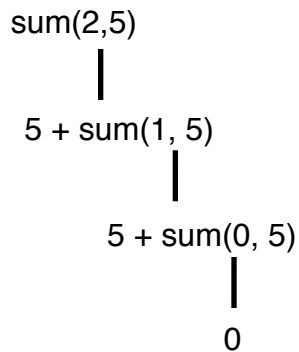


Make an evaluation tree for the method calls below. Also, indicate what the final result is.

```
public int sum(int counter, int x)
{
    if(counter <= 0)
        return 0;
    else
        return x + sum(counter-1, x);
}
```

```
public int sum2(int x, int target)
{
    if(x >= target)
        return x;
    else
        return x + sum2(x+1, target);
}
```



sum(3,6)

sum2(1,4)

sum2(4,1)

Final Result: 10

Final Result:

Final Result:

Final Result:

These two methods print out numbers. One will print out the numbers in ascending order (1, 2, 3, ...) while the other will print in descending order (3, 2, 1...)? **Hint:** try making an evaluation tree for print(1, 3) and print2(1, 3). In general the left side of your tree will happen first

```
public void print(int x, int target)
{
    if(x >= target)
        System.out.println(x);
    else {
        System.out.println(x);
        print(x+1, target);
    }
}
```

```
public void print2(int x, int target)
{
    if(x >= target)
        System.out.println(x);
    else {
        print2(x+1, target);
        System.out.println(x);
    }
}
```

Evaluate these recursive methods for the given inputs. Draw a tree to help you.

```
public double newPop(double popul, double r, double t)
{
    if(t <= 0)
        return popul;
    else
        return newPop(popul*(1+r), r, t-1);
}
```

newPop(100, 0.01, -1)

```
public int popTime(double popul, double r, double goal)
{
    if(popul >= goal)
        return 0;
    else
        return 1 + popTime(popul*(1+r), r, goal);
}
```

newPop(100, 0.01, 5)

popTime(100, 0.2, 200)

```
public double oldPop(double t, double r, double pop)
{
    if(t <= 0)
        return pop;
    else
        return oldPop(t-1, r, pop*(1-r));
}
```

oldPop(5, 0.1, 1000)

```
public int fib(int n)
{
    if(n <= 1)
        return n;
    else
        return fib(n-1) + fib(n-2);
}
```

fib(1)

fib(4)