Instructions: In-class exercises are meant to introduce you to a new topic and provide some practice with the new topic. **Work in a team of up to 4 people to complete this exercise.** You can work simultaneously on the problems, or work separate and then check your answers with each other. **Turn in one copy of the exercise per group**.

Names:

Proofs: Analysis of Algorithms

Function execution

Example
Identify how many times the following code segment gets executed.

```
for i in range( 2 ):
    for j in range( 3 ):
        print( i, j, i*j )
```

Let's step through all the times this executes:

1. $i = 0$
   (a) $j = 0$ Print 0*0, 0.
   (b) $j = 1$ Print 0*1, 0.
   (c) $j = 2$ Print 0*2, 0.
2. $i = 1$
   (a) $j = 0$ Print 1*0, 0.
   (b) $j = 1$ Print 1*1, 1.
   (c) $j = 2$ Print 1*2, 2.

We can see that this executes the internal print statement a total of 6 times. We could also calculate this by taking the outer loop range (0, 1, 2) and multiply it by the inner loop range (0, 1, 2, 3).

Question 1

Identify how many times each the following code segment gets executed.

```
1 for ( int i = 0; i < 5; i++ )
2 {
3     for ( int j = 0; j < 2; j++ )
4     {
5         Output( i * j );
6     }
7 }
```

Question 2

Identify how many times each the following code segment gets executed.

```
1 for ( int i = 0; i < 5; i++ )
2 {
3     for ( int j = i; j < 5; j++ )
4     {
5         Output( i * j );
6     }
7 }
```

Question 3

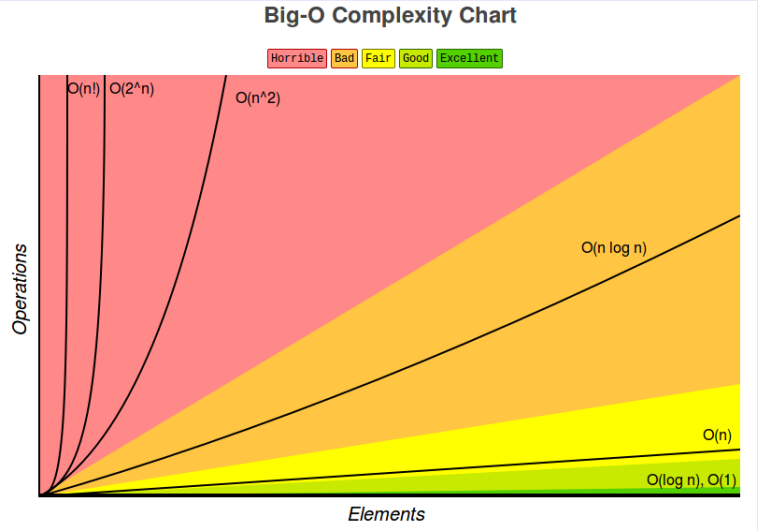
Identify how many times each the following code segment gets executed.

```
1  int i = 0;
2  int j = 20;
3
4  while ( i < j )
5  {
6      i++;
7      j--;
8      Output( i, j );
9  }
```

Basics of identifying algorithm growth rate

Growth Rates

For a growth rate, we aren't interested in the exact amount of times a loop iterates in a function, but instead in the growth rate: As the size n increases, how much does the time execution of the algorithm grow?



From http://bigocheatsheet.com/

Constant, $O(1)$: No loops, no recursion, just one or more commands. The size of n does not affect the execution time.

Linear, $O(n)$: Often a single loop, going over n items. As n grows, the function's execution time grows 1:1.

Quadratic, $O(n^2)$: Nested loops, going over n items over n^2 iterations. As n grows, the function's execution time grows by $n \times n$.

Question 4

Identify the "Big-O" growth rate value for the following.

```
1 def GetMax( a, b ):
2     if ( a > b ):   return a
3     else:           return b
```

Question 5

Identify the "Big-O" growth rate value for the following.

```
1 def GetListMax( myList ):
2     maxVal = myList[0]
3
4     for i in range( 1, len( myList ) ):
5         if myList[i] > maxVal:
6             maxVal = myList[i]
7
8     return maxVal
```

Question 6

Identify the "Big-O" growth rate value for the following.

```
1 def Sum( myList ):
2     sumVal = 0
3
4     for num in myList:
5         sumVal += num
6
7     return sumVal
```

Question 7

Identify the "Big-O" growth rate value for the following.

```
1  void DrawSquare( int width )
2  {
3      for ( int y = 0; y < width; y++ )
4      {
5          for ( int x = 0; x < width; x++ )
6          {
7              cout << "*";
8          }
9          cout << "\n";
10     }
11 }
```

Question 8

Identify the "Big-O" growth rate value for the following.

```
1  bool IsAnagram( string text )
2  {
3      int a = 0;
4      int z = text.size() - 1;
5
6      while ( a < z )
7      {
8          if ( text[a] != text[z] )
9          {
10             return false;
11         }
12         a++;
13         z--;
14     }
15     return true;
16 }
```