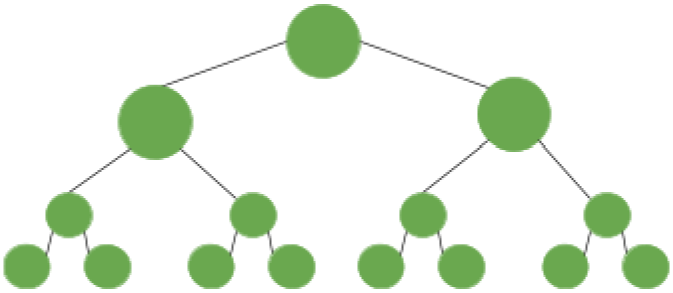
Part 1 Tasks:

1. Recursive statement for factorial funcion N! = N (N-1)!
   The base case is when N==1 when a set variable reaches 1 it will stop (1-1) = 0
   Begin

   ```
       Int factorial(int N)

       If N=0: return result
       If N!=0: return N *  factorial(N-1)
   ```
2.
   a. Iterative: $S(n) = \sum_{i=0}^{n} i$

   b. Recursive: $S(n) = \frac{n(n+1)}{2}$
3. A binary is exactly that; a recursive data structure. If we look at the bigger picture we can see that a binary tree is just a ton of mini binary trees, this makes it so the larger problem can be scaled down into smaller problems to solve. That is why it’s a good structure to store data. There are different types of trees but these characteristics are specifically for the binary tree.



4. Example array

2	6	9	4	1	7	2	5	8	3

The goal is to ignore the last digit over and over again, and sort what remains

1. First, start by sorting an array of n-(n-2) elements, just the first two n-(n-3)....n-(n-i) until i=n
2. The base case is n, when we reach an n number of elements it will stop
3. In order for this to work, we’ll need to do multiple passes.

The first use for loop only to go through all values
Start by comparing element a[n-n] and a[n-(n-1)]
If the first element is larger then swap()
Or else, call the function again to compare a[n-(n-i)] and a[n-(n-i+1)]
Do this until you reach the end of the array
Basically just recursive bubble sort