Hands-On 3

```
function x = f(n)
  x = 1;
  for i = 1:n
     for j = 1:n
        x = x + 1;
```

1. Find the runtime of the algorithm mathematically (I should see summations).

- The outer loop runs n times.
- The inner loop runs n times.
- i.e., the total number of operations is the sum of iterations of both the loops because the inner loop runs n times for each of the outer loop's iterations.

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} (1) = n^2$$

2. Time this function for various n e.g., n = 1,2,3.... You should have small values of n all the way up to large values. Plot "time" vs "n" (time on y-axis and n on x-axis). Also, fit a curve to your data, hint it's a polynomial.

3. Find polynomials that are upper and lower bounds on your curve from #2. From this specify a big-O, a big-Omega, and what big-theta is.

- Big-O (Upper Bound): the worst-case time complexity.
  As the polynomial is quadratic, the upper bound is $O(n^2)$.

- Big-Omega (Lower Bound): the best-case time complexity.
  The quadratic term (n^2) dominates as n grows, so lower bound is $\Omega(n^2)$.

- Big-Theta: if the upper and lower bounds are close, the polynomial represents the actual growth rate of the function.
  As the polynomial fits the data well and grows quadratically, the time complexity is $\Theta(n^2)$.

4. Find the approximate (eye ball it) location of "n_0". Do this by zooming in on your plot and indicating on the plot where n_0 is and why you picked this value. Hint: I should see data that does not follow the trend of the polynomial you determined in #2.

- n = 1000 is where is looks like the data points start to deviate from the polynomial graph (don't fit the polynomial fit anymore).

If I modified the function to be:
x = f(n)

```
x = 1;
y = 1;
for i = 1:n
    for j = 1:n
        x = x + 1;
         y = i + j;
```

5. Will this increate how long it takes the algorithm to run (e.x. you are timing the function like in #2)?

- Yes, the extra line of code will increase the time it takes the algorithm to run. But the increase will be very small and negligible because it's just a constant time increase within the inner loop.

6. Will it effect your results from #1?

- In terms of mathematical notation, the extra line of code doesn't effect the results from #1. Though the time slightly increases, the summations don't get affected because it is still influenced by the nested loops $O(n) = n^2$

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} (1) = n^2$$

7. Implement merge sort, upload your code to github and show/test it on the array [5,2,4,7,1,3,2,6].