Preliminaries and Lecture 1
CS6033 Homework Assignment*1

Due Monday September 17th at 5:00 p.m.
Turn in this assignment as a PDF file on NYU classes
No late assignments accepted

1. You have five algorithms, A1 took $o(n^2)$ steps, A2 took $O(n^2)$ steps, and A3 took $\Theta(n^2)$ steps, A4 took $\Omega(n^2)$ steps, A5 took $\Theta(n)$ steps. You had been given the exact number of steps the algorithm took, but unfortunately you lost the paper you wrote it down on. In your messy office you found the following formulas (you are sure the exact formula for each algorithm is on one of them):

   (a) $7n \cdot \log_2 n + 12 \log_2 \log_2 n$
   (b) $7 \cdot (2^{2 \log_2 n}) + n/2 + 7$
   (c) $(2^{\log_4 n})/3 + 120n + 7$
   (d) $(\log_2 n)^2 + 75$
   (e) $7n!$
   (f) $2^{2 \log_2 n}$
   (g) $2^{\log_4 n}$
   (h) $n + n^2$

   For each algorithm write down all the possible formulas that could be associated with it.

2. Suppose you know that $f(n)$ is $o(g(n))$ and $g(n) = 2n^3$. What else do you know?

   Which of the following are true?

   (a) $f(n)$ is $O(g(n))$ (b) $f(n)$ is $\Theta(g(n))$ (c) $f(n)$ is $\Omega(g(n))$ (d) $f(n)$ is $O(n^2)$

   (e) $f(n)$ is $o(n^2 \log(n))$ (f) $f(n)$ is $O(n^3)$ (g) $f(n)$ is $\Omega(n^3)$ (h) none of these

3. In your book,[1] complete homework problem 1-1 pg 15 excluding determining the answer for $n!$ and $n \log(n)$. Assume 31 days in a month.

*Many of these questions came from outside sources.
[1]Make sure you have the third edition.

4. Does the following algorithm correctly print out all the items that appear more than one time in the array (and prints no other values)? If so, prove it using a *loop invariant*! If not, prove that it does not work by providing an example on which it fails and show how this example is a counterexample.

```
PRINT_DUPLICATES(A)
1) for  i = 1 to A.length
2)     for  j = i to A.length
3)         if A[i] == A[j]
4)             PRINT(A[i])
```

5. Is the following algorithm to compute x^y correct? If so, prove it using a *loop invariant*! If not, prove that it does not work by providing an example on which it fails and show how this example is a counterexample.

```
POW(x, y)
1) e = 1
2) while  y > 0 do
3)     if y is odd then
4)         e = e * x
5)     else
6)         y = ⌊y/2⌋
7)         y = y^2
4) return (e)
```

6. What is the run time of the following questions? Expressing your answer using big-Oh, little-oh, Theta, and Omega.

   (a)

```
GAUSSES_SUMMATION_SERIES_1(n)
1) s = 0
2) for  i = 1 to n
3)     for j = i to n
4)         s = s + 1
5) return s
```

   (b)

```
GAUSSES_SUMMATION_SERIES_2(n)
1) s = 0
2) for  i = 1 to n
3)     s = s + i
4) return s
```

7. Professor T. Ortis, thought the insertion sort presented in class seems rather inefficient.[2] He thought he could improve the run time so it runs *asymptotically faster* if in the while loop (lines 5-7) instead of using a backward scan through the already sorted items to find where

[2]Remember Mergesort took only $n \log n$ time.

the jth item belongs, he instead used used binary search to find where the jth item belongs. Is he right?

8. For the following algorithm: Show what is printed by the following algorithm[3] when called with MAXIMUM$(A, 1, 5)$ where $A = [0, 1, 2, 3, 4]$? Where the function PRINT simple prints its arguments in some appropriate manner.

```
MAXIMUM(A, l, r)
1)  if (r − l == 0)
2)      return  A[r]
3)
4)  lmax = MAXIMUM(A, l, ⌊(l + r)/2⌋)
5)  rmax = MAXIMUM(A, ⌊(l + r)/2⌋ + 1, r)
6)  PRINT(rmax, lmax)
7)  if rmax < lmax
8)      return  lmax
9)  else
10)     return  rmax
```

9. In your space MMO RPG you are about to challenge the last boss.[4] The big space battle is about to start, and you decide to go for quantity over quality and will accept anyone who wants to join your team. To get a large team, you ask your friends and team mates to find others who will join you. This works well (perhaps too well?). Your friends and team mates have sent you thousands of names. You wonder how many names you have received. Occasionally two friends send the same name, etc. How can you efficiently and correctly determine how many people on your team?

   Design an efficient algorithm[5]. Your algorithm must run in time $O(n^2)$ where n is the number of names you have received. State what the running time is of your algorithm in big-Oh, Theta, and Omega notation.

10. (3 bonus points) Think of a *good*[6] exam question for the material covered in Lecture 1

[3] Of course it is a bit silly to write a divide and conquer algorithm to find the maximum item in an array. A simple loop would be much better! But this is a homework problem to test you understand how divide and conquer algorithms work.

[4] There is a bug in the code, the game doesn't scale the last boss based on the number of players!

[5] It is odd, but all the people your friends know have unique names - there are not two "Bob Jones"'. People also kindly send the full name of all the people.

[6] Only well thought out questions will receive the bonus points.