CS 373 SPRING 2016:
HW 2 ALGORITHM ANALYSIS

Assigned: 4/6/2016, **Due:** 4/13/2016 (in class)

1 Computing bigO notation (5pts)

Order the following functions by growth rate. Indicate which functions grow at the same rates.

$N, \sqrt{N}, N^{1.5}, N^2, N \log N, N \log\log N, N \log^2 N, N \log(N^2), 2/N, 2^N, 2^{N/2}, 37, N^2 \log N, N^3$

2 Comparing run-times (12 pts)

For this problem, you will need to write some code in Java. We've provided everything you need to get started in the Java skeleton file located at `http://www.cs.washington.edu/education/courses/cse373/16sp/homework/hw02/HW2Prob2.java`

For each of the following six program fragments:
Give an analysis of the running time. Big-Oh will suffice.
Then, implement the code in Java, and give the running time (in milliseconds) for the several values of n listed in the table below. We've set up the skeleton files to make this easier: Look for an "INSERT YOUR CODE HERE" comment; that is where you will add your code. The skeleton is set up to read the value of n from the command line (e.g. `java HW2Prob2 1000`).
One last note: for some of the code fragments and some values of `n` the run time on modern processors is quite long. For those inputs where the the execution takes over 5 minutes, please stop running the program and write "too long" in the table.

	Big-Oh	n=2000	n=2000000	n=2000000000
1				
2				
3				
4				
5				
6				

Finally, using the completed table above, compare your analysis with the actual running times and discuss.

The six fragments:

1.
```
sum = 0;
for (i=0; i<n; i++)
    sum++;
```

2.
```
sum = 0;
for (i=0; i<n; i++)
    for (j=0; j<n; j++)
        sum++;
```

3.
```
sum = 0;
for (i=0; i<n; i++)
    for (j=0; j<n*n; j++)
        sum++;
```

4.
```
sum = 0;
for (i=0; i<n; i++)
    for (j=0; j<i; j++)
        sum++;
```

5.
```
sum = 0;
for (i=0; i<n; i++)
    for (j=0; j<i*i; j++)
        for (k=0; k<j; k++)
            sum++;
```

6.
```
sum = 0;
for (i=1; i<n; i++)
    for (j=1; j<i*i; j++)
        if (j % i == 0)
            for (k=0; k<j; k++)
                sum++;
```

3 Proving bigO bounds (3 pts)

Show that the function $6n^3 + 30n + 403$ is $O(n^3)$.

You will need to use the formal definition of $O(f(n))$ to do this (see Weiss p29). In other words, find values for c and n_0 such that the definition of Big-Oh holds true as we did with the examples in lecture.

Remember, to prove that an equation exists in bigO for some "relative rate of growth" (Weiss pg. 30), there must be a constant multiple of that growth rate c that is greater than the equation for all $n > n0$. For example, $4n + 10$ is in $O(n)$ because there exists an equation ($5n$) that is larger than $4n + 10$ for any $n > n0$ ($n0$ here is equal to 10).

4 Analysis proof by induction (5 pts)

Given the following recursive search function, prove by induction that it correctly returns 1 if the value `val` is in the array `v` and 0 otherwise. (Hint: try working out all the possibilities for arrays of `size = 1` to get a sense of how your proof should proceed.)

```
int search(v[]: integer array, size: integer, val: integer)
    if (size == 0) return 0;
    else
        if (v[size-1] == val) return 1;
        else return search(v, size-1, val);
```

You will need to provide at least these details in a complete proof:

Basis: The case where `size = 0`

Inductive Hypothesis: Assume...

Inductive Step:

Hint: For the inductive hypothesis, you need to assume the function works for an array of size k and then investigate what happens when the array is of size k+1.