CS161 - Homework 1
Due: Thursday July 2, 5:00pm

1. (5 points) What sorting method does the function SORT use from java.util.Arrays? Make sure to cite your source. Why do you think this is the case?

2. (10 points) The table below contains run times for 6 different algorithms. The input sizes ranged from 1000 to 32000 seen at the top of the table. For each of the algorithms, give the θ complexity of the algorithms based on the running times and include a brief explaination for your answer.

Algorithm	1000	2000	4000	8000	16000	32000
A_1	50	378	3,345	26,300	215,680	1,658,002
A_2	99	110	105	976	103	100
A_3	60	130	237	501	954	1999
A_4	1005	1095	1201	1289	1420	1540
A_5	5	21	84	311	1304	5280
A_6	10	22	50	108	245	533

3. (10 points) Arrange the functions below in ascending order of growth rate. Specifically, if $f(n) = O(g(n))$ then $f(n)$ should be before $g(n)$ in the list. If two functions are asympotically equal, i.e. $f(n) = \Theta(g(n))$ then note this in the list by including all elements in a set. For example, given: $n, \log n$, $n+4$, and n^2 the list would be: $\log n, (n, n+4), n^2$.

$(\frac{3}{2})^n$	7	$5n + 20$	3^{3^n}
n	3^n	$n3^n$	n^5
$n!$	$3^{\sqrt{logn}}$	$n^{\log n}$	$\frac{1}{2}n\log(n+5)$
10^6	n^n	$(3n)^2$	$\log\log n$

4. (15 points) Big O

   (a) (5 points) Show that $15n^3 \log n + 10n^2 + 50 = O(n^3 \log n)$.

(b) (10 points) Show that $\log(n!) = \Theta(n \log n)$. (Hint: To show an upper bound, compare $n!$ with n^n. To show a lower bound, compare it with $(n/2)^{(n/2)}$.)

5. (40 points) For the following problems, write **pseudocode** solutions and state the worse case running time:

   (a) (5 points) Given two lists of numbers A and B of lengths m and n respectively, return the intersection of the lists, i.e. all those numbers in A that also occur in B. You can use procedures that we've discussed in class, but no others (e.g. no hashtables).

   (b) (10 points) Write a function MERGE3 that takes **3** sorted lists and merges them into one list.

   (c) (5 points) Write a new merge sort procedure that uses MERGE3. Calculate the overall runtime of this procedure including the calls to MERGE3.

   (d) (10 points) Given a *sorted* list of integers A[1...n], determine if an entry exists such that $A[i] = i$. If an entry exists, return the index, otherwise, return *null*. (Hint: You can do better than $O(n)$. Think divide-and-conquer.)

   (e) (10 points) In some situations, there is not a natural ordering do the data but we can check equality (e.g. images). Given an array of elements A, we would like to determine if there exists a value that occurs in more than half of the entries of the array. If so, return that value, otherwise, return *null*. Assume you can only check equality of elements in the array which takes time $O(1)$.

Extra Credit

6. (5 points) Find an algorithm for 5e. that is $O(n)$.

Just for fun

7. (1 brownie point) Given two *sorted* arrays A and B of lengths m and n respectively, return the kth smallest element in the union of the two lists. Your runtime should be in terms of both m and n and should not depend on k.