CSCI 3104-Spring 2015: Assignment #1.

**Assigned date:** Friday 1/16/2015,
**Due date:** Thursday, 1/22/2015, before 2:00 PM
**Maximum Points:** 45 points + 5 for legibility + 10 points to be completed online.

**Note:** This assignment must be turned in on paper, before class. Please do not email: it is very hard for us to keep track of email submissions. Further instructions are on the class page: http://csci3104.cs.colorado.edu

**P1 (Ternary Search, 35 points)** Prof. Seeker wishes to search a large array of \( n \) sorted integers for the number 42. Instead of implementing binary search routine, she formulates an advanced search routine called ternary search.

As a member of her advanced analysis team, you are asked to formulate and analyze this algorithm.

Here is the overall idea of ternary search to check if a sorted list \( a \) of size \( n \) has the number \( k \) in it.

- Ternary search divides the list \( a \) into three roughly equal parts of size \( \frac{n}{3} \) (rounded up/down).
- It then looks at the two elements \( a[\lfloor n/3 \rfloor] \) and \( a[2 \times \lfloor n/3 \rfloor] \).
- If \( k < a[\lfloor n/3 \rfloor] \) it searches in the sublist from 0 to \( n/3 \), else if \( a[\lfloor n/3 \rfloor] \leq k < a[2 \times \lfloor n/3 \rfloor] \) search in the sublist from 0 to \( 2 \times n/3 \); otherwise search the the list from \( 2 \times n/3 \) to \( n \).

(A, 10 points online) Complete the idea of ternary search by writing a recursive Python code for ternary search. Write a function \( \text{ternarySearch}(a,k) \) that returns True if \( k \) is found in a nonempty list \( a \); or False otherwise.

Submit the program online through moodle: we have setup a special quiz called “Ternary Search Implementation”. The score on moodle will be automatically added to your grade for this assignment.

(B, 5 points) Demonstrate the working of your program on the function call:

\[
\text{ternarySearch}( [1,3,5,10,12,15,32,91, 125,132], 18).
\]

Specifically list the recursive calls that will take place. Submit this part in paper/pencil

(C, 10 points) Prove the theorem that \( \text{ternarySearch} \) is correct: I.e, if \( k \) is contained in the list \( a \), then it returns True and otherwise it returns False.

Use strong induction. Clearly write down the statement of the theorem, the base case and induction hypothesis.

Your proof does not need to exceed more than 10 sentences (3/4 of a page in total).

(D, 10 points) For an array of size \( n \), derive a tight upper bound on the number of recursive calls made.
If you are on the right track, the answer should have 5 lines or less.

**P2, 20 points** A list of size \( N \) is all but \( k \) sorted if all but \( k \) elements are out of position.

For instance, the list \([1,3,5,2,7,10,8]\) is all but 2 sorted.
(A, 15 points) Prove that the running time of insertion sort algorithm on a given list $a$ of size $N$ that is all but $k$ sorted is bounded by $O(N \times k)$.

(B, 5 points) Argue why the recursive version of mergeSort presented in class can still take $O(n \log n)$ time even if it’s input is already a fully sorted array? How can this be fixed?