Student Name: ______________________________ J#: ________________

CSC 323 Algorithm Design and Analysis
Fall 2016
Instructor: Dr. Natarajan Meghanathan

Quiz 4 and Bonus Quiz # 1 (Take Home)

Total: 50 points (Quiz 4)

Due: October 11, 2016 for both Quiz 4 and Bonus Quiz # 1 (11.30 AM, in class). Quiz 4 solutions and Bonus Quiz # 1 solutions submitted after 11.30 AM will not be accepted. Submit a printed hardcopy in class (with this quiz sheet as a cover page and your name and J# on the top of the sheet).

Note: Strictly, there should NOT be any copying. If the instructor finds that two or more quiz solutions involve some sort of copying, all the concerned students found to be involved in copying will get a zero.

Recall the problem of finding local minimum in an array. An element is a local minimum if it is less than the element to its immediate left as well as less than the element to its immediate right. That is, we say the element A[i] at index i is a local minimum only if A[i] < A[i-1] and A[i] < A[i+1]. In this sense, an array A[0...n-1] can possibly have local minimum only at index values 1, ..., n-2 and not at indexes 0 and n-1.

In class, we discussed the pre-requisites for an array to have at least one local minimum. Recall the pre-requisites, as listed below:

(1) The array should have at least 3 elements
(2) The first two elements of the array should be decreasing and the last two elements of the array should be increasing
(3) The array should have distinct elements

Q1 (25 pts): In the slides, I did not give a formal pseudo code to find a local minimum (given the above three pre-requisites). Provide a pseudo code to find a local minimum in the array. (Note that: it is sufficient to find just one local minimum in the array). Create an array of 10 distinct integers that also satisfies the other two pre-requisites as listed above and show the execution of your pseudo code of Q1 to find a local minimum.

Q2 (25 pts): Extend the pseudo code of Q1 to find as many local minimum as possible in an array that still satisfies the above three pre-requisites. Note that you may not be able to find all the local minimum in an array; but, your pseudo code should strive to find as many local minimum as possible.

[Hint: Replace each of the local minimum that you find with a distinct negative integer that is smaller than the rest of the integers in the array, and continue the binary search in the remaining search space]

Show that the time-complexity of your algorithm/pseudo code now is still Θ(logn).

Illustrate the working of your pseudo code to find as many local minimum as possible on the following array.

Index	0	1	2	3	4	5	6	7
Array, A	8	5	7	2	3	0	1	9

Student Name: ______________________________ J#: ________________

BONUS QUIZ # 1
(equivalent to 3% bonus in the final overall score)

Note that we could determine all the local minimum in an array by simply traversing the array from left to right (i.e., from index 1 to n-2 for an array A[0, ..., n-1]). But, that would be a Θ(n) linear search algorithm.

The binary search-based algorithm/pseudo code for Q2 is of time-complexity Θ(logn), but is not guaranteed to detect all the local minimum in an array.

Let us define the effectiveness of the binary search-based algorithm for an array A as the ratio of the number of local minimum determined by the algorithm on the array A to that of the number of local minimum determined by a linear search algorithm.

Implement the binary search-based algorithm for Q2 as well as the linear search algorithm for finding as many local minimum in an array as possible.

For any given array size 'n', you need to be able to generate arrays of unique positive integers, and reset the integers at index 0 and index n-1 to a significantly larger value (say 10001 and 10002) if they are smaller than the integers to their immediate right and left respectively. The rest of the integers in your array should be of values between 1 to 10000.

For simulation purposes: vary the array size (n) with values such as: 64, 128, 256, 512, 1024 and 2048. For each array size, run 50 trials and average the results for the effectiveness of the binary search algorithm (as defined above). Also, determine the average values for the run-times of your binary search and linear search algorithms, and their ratio.

Plot the results of (i) the array size, n, vs. the average effectiveness of the binary search algorithm
(ii) the array size n, vs. the ratio of the average values for the run-times of your binary search algorithm and the linear search algorithm

What to submit:
(i) Your program code for the binary search algorithm corresponding to the pseudo code of Q2.
(ii) Your program code for the linear search algorithm
(iii) Snapshots of the outputs (average effectiveness and ratio of the average run-times) for arrays of size 64 and 2048.
(iv) Excel plots of your results, as mentioned above.

Helpful Videos
You may find the videos here to be useful:
Basics of Vector class: https://youtu.be/vbySmVOjLIk
Creating Random Elements (for 1-dim and 2-dim arrays): https://youtu.be/J0A3qrfiq38