Computer Science C73 September 22, 2021
Scarborough Campus University of Toronto

Homework Assignment #3
(worth 4% of the course grade)
Due: September 29, 2021, by 11:59 pm

• ***You must submit your assignment through the Crowdmark system.*** *You will receive by email an invitation through which you can submit your work in the form of separate PDF docucments with your answers to each question of the assignment. To work with a partner, you and your partner must form a group on Crowdmark. Crowdmark does not enforce a limit on the size of groups.* ***The course policy that limits the size of each group to at most two remains in effect:*** *submissions by groups of more than two persons will not be graded.*
• *It is your responsibility to ensure that the PDF files you submit are legible. To this end, I encourage you to learn and use the LaTex typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You are not required to produce the PDF files you submit using LaTex; you may produce it any way you wish, as long as the resulting document is legible.*
• *By virtue of submitting this assignment you (and your partner, if you have one) acknowledge that you are aware of the policy on homework collaboration for this course.*[a]
• *For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbooks, by referring to it.*
• *Unless we explicitly state otherwise, you may describe algorithms in high-level pseudocode or point-form English, whichever leads to a clearer and simpler description.*
• *Unless we explicitly state otherwise, you should justify your answers. Your paper will be graded based on the correctness and efficiency of your answers, and the clarity, precision, and conciseness of your presentation.*

[a] "In each homework assignment you may collaborate with at most one other student who is currently taking CSCC73. If you collaborate with another student on an assignment, you and your partner must submit only one copy of your solution, with both of your names. The solution will be graded in the usual way and both partners will receive the same mark. Collaboration involving more than two students is not allowed. **For help with your homework you may consult only the instructor, TAs, your homework partner (if you have one), your textbook, and your class notes. You may not consult any other source.**"

Question 1. (10 marks) Consider the algorithm WeirdSort(A, ℓ, r) below, where $A[1..n]$ is an array, and $1 \leq \ell \leq r \leq n$. This is a recursive sorting algorithm that (unlike Mergesort) requires no merging and (unlike Quicksort) requires no partitioning.

```
    WeirdSort($A, \ell, r$)
1   if $r = \ell$ then                          ▷ $A[\ell..r]$ has length 1
2       return
3   elsif $r = \ell + 1$ then                   ▷ $A[\ell..r]$ has length 2
4       if $A[\ell] > A[r]$ then swap $A[\ell]$ and $A[r]$
5       return
6   else                                        ▷ $A[\ell..r]$ has length at least 3
7       $t := \lceil 2(r - \ell + 1)/3 \rceil$              ▷ $t$ is two-thirds of the length of $A[\ell..r]$
8       WeirdSort($A, \ell, \ell + t - 1$)          ▷ sort the first two-thirds of $A[\ell..r]$
9       WeirdSort($A, r - t + 1, r$)          ▷ sort the last two-thirds of $A[\ell..r]$
10      WeirdSort($A, \ell, \ell + t - 1$)         ▷ re-sort the first two-thirds of $A[\ell..r]$
11      return
```

a. Prove that a call to WEIRDSORT(A, ℓ, r) will sort the subarray $A[\ell..r]$ in increasing order. To simplify matters, you may assume that the length $r - \ell + 1$ of $A[\ell..r]$ is a power of 3.

b. Use the master theorem to determine the running time $T(n)$ of the call WEIRDSORT$(A, 1, n)$.

c. How does the running time of WEIRDSORT$(A, 1, n)$ compare to the running time of Bubblesort on array $A[1..n]$? Justify your answer.

Question 2. (15 marks) A ***local minimum*** of a non-empty array A containing distinct numbers is an element of A that is smaller than all of its neighours. (Every element of A except the first and last has two neighbours; the first and last elements have one neighbour each, unless A has only one element. In that case, the single element of A has no neighours and is therefore, by definition, a local minimum.)

Design an algorithm that, given an array $A[1..n]$ containing $n > 1$ distinct numbers, finds a local minimum in $O(\log n)$ time. Justify the correctness and running time of your algorithm.