CSC263H1: Problem Set 1

Due Tuesday September 17 before 10pm

General instructions

Please read the following instructions carefully before starting the problem set. They contain important information about general problem set expectations, problem set submission instructions, and reminders of course policies.

- Your problem sets are graded on both correctness and clarity of communication. Solutions that are technically correct but poorly written will not receive full marks. Please read over your solutions carefully before submitting them.
- Each problem set may be completed in groups of up to three. If you are working in a group for this problem set, please consult the articles on collaboration and plagiarism on posted on quercus.
- Solutions must be typeset electronically, and submitted as a PDF with the correct filename. **Handwritten submissions will receive a grade of ZERO.**

  The required filename for this problem set is **problem_set1.pdf**.
- Problem sets must be submitted online through CrowdMark. If you haven't used CrowdMark before, give yourself plenty of time to figure it out, and ask for help if you need it! If you are working with a partner, you must form a group on CrowdMark, and make one submission per group. "I didn't know how to use CrowdMark" is not a valid excuse for submitting late work.
- Your submitted file(s) should not be larger than 9MB. You might exceed this limit if you use a word processor like Microsoft Word to create a PDF; if it does, you should look into PDF compression tools to make your PDF smaller, although please make sure that your PDF is still legible before submitting!
- Submissions must be made *before* the due date on CrowdMark.
- The work you submit must be that of your group; you may not use or copy from the work of other groups, or external sources like websites or textbooks.

Additional instructions

- None.

1. **[20 marks] Complexity**

Consider these procedures:

```
Sum(input array $A$) returns **Integer**
    $total = 0$
    **for** $i = 1$ **to** $A.size$
        $total \leftarrow total + A[i]$
    **return** $total$

CheckArray(input array $A$) returns **Boolean**
    **if** $A.size < 3$ **then**
        **return** TRUE
    **if** $A[1] = 0$ **then**
        **return** FALSE
    **for** $i = 1$ **to** $A.size - 2$
        **if** Sum($A[i : i+2]$) $\neq 0$ **then**
        **return** FALSE
    **return** TRUE
```

In the code above, the left arrow denotes the assignment operation, i.e. $x \leftarrow x + 1$ increments x; **return** exits a function immediately. The array A is an *input* to the function and its scope is *local* to the function. The indices of array A start at 1. The colon denotes the subarray operation, i.e., $A[j : k]$ is a constant-time operation on A that returns the subarray consisting of the elements $A[j], A[j+1], \ldots A[k]$, when $j \leq k$, and the empty array when $j > k$.

Assume that each assignment, comparison and arithmetic operation takes *one unit* of computational cost.

(a) Let $t_{\text{Sum}}(A)$ be the cost of executing Sum on an input A. Express $t_{\text{Sum}}(A)$ in terms of the size of A. Justify your analysis of the computational cost.

(b) Let $T(n)$ be the worst-case cost of executing CheckArray on an array A of size n. Express $T(n)$ in terms of n. Fully justify your analysis of the computational cost.

(c) Give a function $f(n)$ such that $T(n) \in \Theta(f(n))$. Justify your answer by explaining why $T(n) \in O(f(n))$ and $T(n) \in \Omega(f(n))$. A sound and clear justification is required to receive credit.

2. **[20 marks] List ADT.** Consider the following abstract data type that we will call a "List."

Objects: A ordered multiset L of integers. Note that a multiset allows for the repitition of elements, so $[1]$ and $[1, 1]$ are distinct lists. In this case, "ordered" means that distinct lists may have distinct orders. I.e. $[-1, 0, 1]$ and $[1, 0, -1]$ are distinct lists; it does not mean that the contents must be in any particular order (e.g. non-decreasing).

Operations:

- Insert(L, x, i): Insert the integer x into L at position i.
- Pop(L, i): Remove and return the integer at position i in L.
- Remove(L, x): Delete the first occurence of the integer x from L.
- Size(L): Return the number of integers in L.
- Sort(L): Modify L so that the integers are in non-decreasing order.

(a) Give two different implementations of the list ADT; one using a linked data structure, and one using fixed length arrays.

Clarification (September 12): Here you should describe the ways in which you will represent data. You will describe how operations work in part (b).

(b) Describe in detail (pseudocode) how each operation above works for each of your implementations of the list ADT.

(c) Describe one situation (pseudocode) for each of your implementations where it would be preferable to the other, and justify your choice based on the running times of the different implementations.