Computer Science CSC263H — January 14, 2016
St. George Campus — University of Toronto

Homework Assignment #1
Due: January 28, 2016, by 5:30 pm

- You must submit your assignment as a PDF file of a typed (**not** handwritten) document through the MarkUs system by logging in with your CDF account at `markus.cdf.toronto.edu/csc263-2016-01`. To work with a partner, you and your partner must form a group on MarkUs.
- The PDF file that you submit must be clearly legible. To this end, we encourage you to learn and use the LaTex typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You can use other typesetting systems if you prefer, but handwritten documents are not accepted.
- 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 textbook, by referring to it.
- Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness 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 one of the sections of CSC263H taught this term. 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 course instructors, teaching assistants, your homework partner (if you have one), your textbook and your class notes. You may not consult any other source.**"

Question 1. (16 marks)

In the following procedure, the input is an array $A[1..n]$ of arbitrary integers, $A.size$ is a variable containing the size n of the array A (assume that A contains at least $n > 2$ elements), and "return" means return from the procedure call.

```
0. Procedure nothing(A)
1.    A[1] := 0
2.    A[2] := 1
3.    for i = 3 to A.size do
4.        s := 0
5.        for j = 1 to i-1 do s := s + A[j]
6.        if A[i] is not equal to s then return
7.    return
```

Let $T(n)$ be the worst-case time complexity of executing procedure **nothing()** on an array A of size $n > 2$. Assume that assignments, comparisons and arithmetic operations, like additions, take a constant amount of time each.

a. (4 marks) State whether $T(n)$ is $O(n^2)$ and justify your answer.

b. (12 marks) State whether $T(n)$ is $\Omega(n^2)$ and justify your answer.

Any answer without a sound and clear justification will receive no credit.

Question 2. (14 marks) Let A be an array containing n integers. Section 6.3 of our textbook (CLRS) describes a procedure, called BUILD-MAX-HEAP(A), that transforms array A into a max-heap in $O(n)$ time. That procedure works "bottom-up", using MAX-HEAPIFY repeatedly.

Another way of transforming A into a max-heap is to insert the elements of A into the heap one at a time. Specifically, the algorithm is as follows:

```
BUILD-BY-INSERTS($A$)
    $A.heap\text{-}size := 1$
    for $i := 2..n$ do
        MAX-HEAP-INSERT($A, A[i]$)
```

a. (4 marks) Give an example of an input array A for which the two procedures BUILD-MAX-HEAP and BUILD-BY-INSERTS produce different outputs. Keep your example as small as possible.

b. (10 marks) Let $T(n)$ be the worst-case time complexity of BUILD-BY-INSERTS for an input array A of size n. Prove that $T(n)$ is $\Theta(n \log n)$. (Recall that the worst-case time complexity of BUILD-MAX-HEAP is $O(n)$, and therefore BUILD-MAX-HEAP is more efficient than BUILD-BY-INSERTS.)

Question 3. (20 marks)

Let I_n be the set of n integers $\{1, 2, \ldots, n\}$ where n is some power of 2.

Note that we can easily use an n-bit vector (i.e., an array of n bits) $B[1..n]$ to maintain a subset S of I_n and perform the following three operations (where j is any integer in I_n) in constant time each:

INSERT(j): insert integer j into S.

DELETE(j): delete integer j from S.

MEMBER(j): return **true** if $j \in S$, otherwise return **false**.

Describe a data structure that supports all the above operations ***and*** also the following operation

MAXIMUM: return the greatest integer in S

such that:

- The worst-case time complexity of operations INSERT(j), DELETE(j), and MAXIMUM is $O(\log n)$ each. The worst-case time complexity of MEMBER(j) is $O(1)$.
- The data structure uses only $O(n)$ bits of storage.

  Note that the binary representation of an integer i where $1 \leq i \leq n$ takes $\Theta(\log n)$ bits. Assume that any pointer also takes $\Theta(\log n)$ bits.

A solution that does not meet ***all*** the above requirements may not get any credit.

HINT: Complete binary trees can be implemented without using pointers.

a. (6 marks) Describe your data structure by drawing it for $n = 8$ and $S = \{1, 2, 6\}$, and by explaining this drawing briefly and clearly. Your drawing must be very clear.

b. (10 marks) Explain how the operations INSERT(j) and MAXIMUM are executed, and why they take $O(\log n)$ time in the worst-case. Be brief and precise.

c. (4 marks) Explain how the operation MEMBER(j) is executed, and why it takes $O(1)$ time in the worst-case. Be brief and precise.