Homework Assignment #2
Due: February 11, 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. (10 marks)

In this question, you must use the insertion and deletion algorithms as described in the "Balanced Search Trees: AVL trees" handout posted on the course web site.

Insert into an initially empty AVL tree each of the following keys, in the order in which they appear in the sequence: 0, 25, 19, 5, -2, 28, 13, -5, 2, 6, 14, 7.

Show the resulting AVL tree T. (Only the final tree should be shown; any intermediate trees shown will be disregarded, and not given partial credit.)

From AVL tree T, delete 2, and show the resulting tree.

In the two trees you must show the key and balance factor of each node.

Question 2. (20 marks)

In the following, B_1 and B_2 are two binary search trees such that every key in B_1 is smaller than every key in B_2.

Describe an algorithm that, given pointers b_1 and b_2 to the roots of B_1 and B_2, merges B_1 and B_2 into a single binary search tree T. Your algorithm should satisfy the following two properties:

1. Its worst–case running time is $O(min\{h_1, h_2\})$, where h_1 and h_2 are the heights of B_1 and B_2.
2. The height of the merged tree T is at most $max\{h_1, h_2\} + 1$.

Note that the heights h_1 and h_2 are ***not*** given to the algorithm (in other words, the algorithm does not "know" the heights of B_1 and B_2). Note also that B_1, B_2 and T are ***not*** required to be balanced.

Describe your algorithm, and justify its correctness and worst-case running time, in ***clear and concise*** English.

NOTE: Partial credit may be given for an algorithm that runs in $O(max\{h_1, h_2\})$ time.

Question 3. (24 marks)

The task in this question is to compute the medians of all prefixes of an array. As input we are given the array $A[1\,..\,n]$ of arbitrary integers. Using a heap data structure, design an algorithm that outputs another array $M[1\,..\,n]$, so that $M[i]$ is equal to the median of the numbers in the subarray $A[1\,..\,i]$. Recall that when i is odd, the median of $A[1\,..\,i]$ is the element of rank $(i+1)/2$ in the subarray, and when i is even, the median is the average of the elements with ranks $i/2$ and $i/2+1$. Your algorithm should run in worst-case time $O(n \log n)$.

a. (20 marks) Describe your algorithm in clear and concise English, and also provide the corresponding pseudocode. Argue that your algorithm is correct.

b. (4 marks) Justify why your algorithm runs in time $O(n \log n)$.

[***The questions below will not be corrected/graded. They are given here as interesting problems that use material that you learned in class.***]

Question 4. (0 marks) This question is about the cost of successively inserting k elements into a binomial heap of size n.

a. Prove that a binomial heap with n elements has exactly $n - \alpha(n)$ edges, where $\alpha(n)$ is the number of 1's in the binary representation of n.

b. Consider the worst-case total cost of successively inserting k new elements into a binomial heap H of size $|H| = n$. In this question, we measure the worst-case cost of inserting a new element into H as the maximum number of pairwise comparisons between the keys of the binomial heap that is required to do this insertion. It is clear that for $k = 1$ (i.e., inserting one element) the worst-case cost is $O(\log n)$. Show that when $k > \log n$, the *average* cost of an insertion, i.e., the worst-case total cost of the k successive insertions divided by k, is bounded above by constant.

Hint: Note that the cost of each one of the k consecutive insertions varies — some can be expensive, other are cheaper. Relate the cost of each insertion, i.e., the number of key comparisons that it requires, with the number of extra edges that it forms in H. Then use part (a).