Computer Science B63 — January 17, 2007
Scarborough Campus — University of Toronto

Homework Assignment #1
Due: January 31, 2007, by 1:10 pm

1. *Please complete and attach (with a staple) an assignment cover page to the front of your assignment. You may work alone or with one other student. If you work in a group, write both your names on the cover sheet and submit only one copy of your homework.*
2. *If you do not know the answer to a question (including the programming question), and you write "I (We) do not know the answer to this question", you will receive 20% of the marks of that question. If you just leave a question blank with no such statement, you get 0 marks for that question.*
3. *Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision and conciseness of your presentation.*

Question 1. (20 marks) In class we studied *binary* heaps, i.e., heaps that store the elements in nearly-complete *binary* trees. This question is about *ternary* heaps, i.e., heaps that store the elements in nearly-complete *ternary* trees (where each node has at most *three* children, every level is full except for the bottom level, and all the nodes at the bottom level are as far to the left as possible). Here we focus on MAX heaps, where the priority of each node in the ternary tree is greater or equal to the priority of its children (if any).

a. (3 marks) Explain how to implement a ternary heap as an array A with an associated $Heapsize$ variable. Specifically, explain how to map each element of the tree into the array, and how to go from a node to its parent and to each of its children (if any).

b. (4 marks) Suppose that the ternary heap contains n elements.
(1) What elements of array A represent internal nodes of the tree?
(2) What is the height of the tree?

c. (13 marks) Consider the following operations on a ternary heap represented as an array A.

- INSERT(A, key): Insert key into A.
- EXTRACT_MAX(A): Remove a key with highest priority from A.
- UPDATE(A, i, key), where $1 \leq i \leq heapsize(A)$: Change the priority of $A[i]$ to key and restore the heap ordering property.
- REMOVE(A, i), where $1 \leq i \leq heapsize(A)$: Delete $A[i]$ from the heap.

For each one of these four operations, describe an efficient algorithm to implement the operation, and give the worst-case time complexity of your algorithm for a heap of size n. Describe your algorithm using high-level pseudo-code similar to that used in your textbook, with clear explanations in English. Express the worst-case time complexity of your algorithm in terms of Θ and justify your answer.

Question 2. (20 marks) This question is about the worst-case cost of successively inserting k elements into a binomial heap (*alias* binomial queue) of size n.

a. (8 marks) 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. (12 marks) 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 elements of the binomial heap that is required to do this insertion. In class we proved that for $k = 1$ (i.e., inserting one element) the worst-case cost is

$O(\log n)$. Show that if $k > \log n$, then the worst-case *total* cost of successively inserting k elements into H is only $O(k)$. In other words, if $k > \log n$ then the *average* cost of an insertion (i.e., the worst-case total cost divided by k) is bounded by a *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 pairwise comparisons that it requires, with the number of extra edges that it forms. Then use part (a).

Question 3. (20 marks) This question is a programming assignment. To see its description follow the link given in the "Assignments" section of the course web page.