Computer Science 263 October 16, 2007
St. George Campus University of Toronto

Homework Assignment #2
Due: October 25, 2007, by 10:10 am

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, 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. *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.*

Question 1. (10 marks) In this question, you must use the insertion and deletion algorithms as described in the "Notes on AVL trees" handout posted on the course web site.

a. (5 marks) Insert into an initially empty AVL tree each of the following keys, in the order in which they appear in the sequence: 10, 20, 22, 7, 18, 13, 16, 8, 2, 15, 4, 1, 14.

Show the resulting AVL tree T, including the key and balance factor of each node. (Only the final tree should be shown; any intermediate trees shown will be disregarded, and not given partial credit.)

b. (5 marks) From the AVL tree T you built above, delete nodes 7 and 1 (in this order), and show the resulting tree. Show the key and balance factor of each node. (Show only the final tree.)

Question 2. (25 marks) A large running club is in need of a new data structure to store its runners' results. Each record consists of a unique integer identifier (composed from the runner's name, run date and run course), the runner's age, and the runner's time. The data structure must support insert and delete operations, queries asking for a particular runner's time, and queries asking for the fastest time run in a particular age group (consisting of runners of a particular age and older). And since the runners are in a hurry, results should be processed or returned in no more than $O(\log n)$ time, where n is the number of records stored.

Your job is to design a data structure D based on AVL trees to support the following four operations:

- INSERT(D, r, a, t): Insert into D a record for identifier r with age a and time t. (Assume that there is no previous record in D for r.)
- DELETE(D, r): Remove the record for runner/course r. If there is no record in D with identifier r, do nothing.
- TIME(D, r): Return the time t for the record in D with identifier r. If there is no record in D with identifier r, return NIL.
- FASTEST(D, a): Return the lowest time attained by a runner in D whose age is real number a or greater. If there is no runner in D of age at least a, return NIL.

Note that the identifiers r are integers (which may come from a very large range, say 64-bit integers) and that the ages a are real numbers (that is, age is not just a whole number of years, but could also include months and days).

a. (6 marks) Give a precise and full description of a data structure that implements this ADT. Your data structure must be based on an AVL tree where each node in the tree represents one record. Illustrate your data structure by giving an example of it on some collection of operations of your choice.

b. (19 marks) For each of the above operations, describe how to implement it in $O(\log n)$ worst-case running time and explain why, in each case, your algorithm achieves this time complexity. You do not need to describe any operations or repeat any complexity analysis that were given in class or in the textbook: simply refer to these as needed (and concentrate on any modifications you require).

Question 3. (6 marks) A large university want to determine the median GPA of its students. The GPA is recorded as a number between 0.00 and 4.00 (recorded to two decimal places). Using a data structure studied in class, design an algorithm to find the median GPA of n students in $\Theta(n)$ time and constant space (you may assume that the number n can be stored using constant space). Briefly justify your algorithm's correctness and complexity.

(Recall that the median in a list is the middle number if the list was ordered. For example, the median of 5,17,12,5,8,4,8 is 8. If there is an even number of elements, either middle element is acceptable.)

Question 4. (14 marks) Consider a hash table T that contains m slots and uses chaining to resolve collisions (where elements are inserted at the head of the linked list). Suppose we have inserted n elements into T, where n is even. In this question you will compute the *exact* expected cost of searching for a key is present in T.

a. (6 marks) Suppose the key k we search for is twice as likely to be one of the first $\frac{n}{2}$ keys that were inserted in T as being one of the last $\frac{n}{2}$ keys inserted. What is the *exact* expected number of keys examined while searching for key k in T? Be sure to explicitly state any assumptions you require, and simplify your answer.

b. (6 marks) Suppose the key k we search for is twice as likely to be one of the last $\frac{n}{2}$ keys that were inserted in T as being one of the first $\frac{n}{2}$ keys inserted. What is the *exact* expected number of keys examined while searching for key k in T? Be sure to explicitly state any assumptions you require, and simplify your answer.

c. (2 marks) What is the difference (in terms of the load factor α, and possibly n or m) between the expected cost of a successful search when the the searched key is selected uniformly (as we saw in class) and the expected costs for (a) and (b)? Are these differences equal (in absolute value) or not? Give a short justification why these differences should or should not be equal.

Question 5. (25 marks) This question is a programming assignment. To see its description follow the link given in the "Assignments" section of the course web page.