Computer Science CSC263H
St. George Campus

February 9, 2017
University of Toronto

Homework Assignment #3

Due: March 2, 2017, 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.teach.cs.toronto.edu/csc263-2017-01`

  To work with one or two partners, you and your partner(s) must form a group on MarkUs.

- If this assignment is submitted by a group of two or three students, the PDF file that you submit should contain for each assignment question:

  1. The name of the student who *wrote* the solution to this question, and
  2. The name(s) of the student(s) who *read* this solution to verify its clarity and correctness.

- 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 partners, if you have any) acknowledge that you are aware of the homework collaboration policy that is stated in the csc263 course web page: `http://www.cs.toronto.edu/~sam/teaching/263/#HomeworkCollaboration`.

- 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.

Question 1. (25 marks)

A *Scheduler* $\mathcal{S}$ consists of a set of *threads*; each thread is a tuple $t = (id, status)$ where id is a distinct positive integer and $status \in \{A, R, S\}$; intuitively, A means active, R means ready to be scheduled, and S means stalled. The operations that Scheduler $\mathcal{S}$ supports are:

NewThread(t): Given a thread $t = (id, status)$, add thread t to $\mathcal{S}$. You can assume that the id of t is different from the id of any thread currently in $\mathcal{S}$ (so that all the threads in $\mathcal{S}$ have distinct ids).

Find(i): If $\mathcal{S}$ has a thread $t = (i, -)$ then return t, else return -1.

Completed(i): If $\mathcal{S}$ has a thread $t = (i, -)$ then remove t from $\mathcal{S}$, else return -1.

ChangeStatus($i, stat$): If $\mathcal{S}$ has a thread $t = (i, -)$ then set the *status* of t to *stat*, else return -1.

ScheduleNext: Find the thread t with smallest id among all the threads whose *status* is R in $\mathcal{S}$, set the *status* of t to A and return t; if $\mathcal{S}$ does not have a thread whose *status* is R then return -1.

In this question, you must describe how to implement the Scheduler $\mathcal{S}$ described above using a *single augmented AVL tree* such that each operation takes $O(\log n)$ time in the worst-case, where n is the number of threads in $\mathcal{S}$. Since this implementation is based on a data structure and algorithms described in class and in the AVL handout, you should focus on describing the extensions and modifications needed here.

a. (5 marks) Give a precise and full description of your data structure. In particular, specify what data is associated with each node, specify what the key is at each node, and specify what the auxiliary information is at each node. Illustrate this data structure by giving an example of it (with a small set of threads of your own choice).

b. (20 marks) Describe the algorithm that implements each one of the five operations above, and explain why each one takes $O(\log n)$ time in the worst-case. *Your description and explanation should be in clear and concise English.* For the operations ChangeStatus($i, stat$) and ScheduleNext, you should also give the algorithm's high-level pseudocode.

Question 2. (15 marks) The *Power of Two Choices* hashing scheme is a method of hashing that reduces the number of collisions when searching in a hash table. Suppose we have a hash table T with m slots and two *independent* hash functions h_1 and h_2. Each hash function satisfies the *Simple Uniform Hashing Assumption* (SUHA): every key in the universe U hashes with equal probability to any hash slot of T, independently from the hash values of all other keys in U. When we insert a new element x, we add it to the chain in one of the two hash slots $T[h_1(x)]$ or $T[h_2(x)]$; we pick the slot in which the chain is shorter. If the chains are same length, we add x to the chain in $T[h_1(x)]$.

a. (5 marks) If the number of empty slots in T is k before inserting x, what is the probability that x is inserted into an empty slot? Justify your answer.

b. (10 marks) Suppose that $m = 4$, and $T[0]$ contains 6 elements, $T[1]$ contains 3 elements, and $T[2]$, $T[3]$ contain 9 elements each. What is the *expected* length of the chain x is inserted into, *not counting* x itself? Justify your answer.

Question 3. (25 marks) You are to write an efficient algorithm for the following problem. Your algorithm is given a sequence (of finite, unknown length) of English words, one at a time. Since we can't account for misspellings, slang, colloquialisms, etc., the universe of words is infinite but assume that the number of distinct words in our sequence is bounded above by a known constant l. A *query* operation can occur at any point between any two inputs in the sequence. When a *query* occurs, the algorithm must return, in sorted order, the most frequent word beginning with each letter. That is, it must return the most frequent word beginning with "a", the most frequent word beginning with "b", etc. Ties in frequency can be resolved by

taking the word occurring first in alphabetical order. If no words beginning with a particular letter have been input, simply return `null` for that letter. Assume words are all lowercase.

Describe a simple algorithm that solves the above problem with the following time complexity:

- $\Theta(1)$ *expected time* to process each input, under some ***reasonable assumptions*** that you should state
- $\Theta(1)$ *worst-case* to perform each *query* operation.

To answer this question, you must:

(1) State which data structure(s) you are using, the items contained within, and any assumptions you are making.

(2) Explain your algorithm ***clearly*** and ***concisely***, in English.

(3) Give the algorithm's ***pseudo-code***, including the code to process an input word and the code for *query*.

(4) Explain why your algorithm achieves the required time complexity described above, given the assumptions in (1).

[***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) Consider an abstract data type that consists of a *set* S of integers on which the following operations can be performed:

Add(i) Adds the integer i to S. If this integer already is in S, then S does not change

Sum(t) Returns the sum of all elements of S that are less than or equal to the integer t. If all the elements of S are greater than t, then return 0.

Describe how to implement this abstract data type using an augmented AVL tree. Each operation should run in $O(\log n)$ worst-case time, where $n = |S|$. Since this implementation is based on a data structure and algorithms described in class and in the AVL handout, you should focus on describing the extensions and modifications needed here.

a. Give a precise and full description of your data structure. In particular, specify what data is associated with each node, specify what the key is at each node, and specify what the auxiliary information is at each node. In particular, what is (are) the augmented field(s), and what identity should this (these) fields satisfy. Illustrate this data structure by giving an example of it (with a small set of your own choice).

b. Describe the algorithm that implements each one of the two operations above, and explain why each one takes $O(\log n)$ time in the worst-case. *Your description and explanation should be in clear and concise English.* For the operations SUM, you should also give the algorithm's high-level pseudocode.

Question 5. (0 marks)

Let $p_2, \ldots, p_n$ be real numbers in $[0, 1]$, to be specified later. Consider the following randomized algorithm:

```
1  x = 1
2  for i = 2 to n
3        With probability p_i, x = i; otherwise x is unchanged.
4  return x
```

a. Consider the value of x returned by the above procedure. Compute $\Pr[x = i]$ for each $1 \leq i \leq n$, under the assumption that $p_2 = p_3 = \ldots = p_n = 1/2$. Give a precise mathematical derivation of these probabilities.

b. Give values for $p_2, \ldots, p_n$ so that for any $1 \leq i \leq n$, $\Pr[x = i] = 1/n$, i.e. x is a uniform sample from $1, \ldots, n$. Prove that, with the values of $p_2, \ldots, p_n$ that you give, the probability of $x = i$ is $1/n$, as required.

Question 6. (0 marks) Assume you have a biased coin, which, when flipped, falls on Heads with probability p, where $0 < p < 1$, and on Tails with probability $1 - p$. However, you do not know p. How can you use the coin to simulate an unbiased coin? Formally, you have access to a procedure FLIPBIASEDCOIN(), which returns either 1 or 0 at random. FLIPBIASEDCOIN() returns 1 with probability p, and 0 with probability $1 - p$, but you do not know p. Design an algorithm that, without making any other random choices except calling FLIPBIASEDCOIN(), returns 1 with probability 1/2 and 0 with probability 1/2. Your algorithm should not use p. You can assume that each time you call FLIPBIASEDCOIN(), the value it returns is independent of all other calls to it.

a. Describe the algorithm in clear and concise English, and prove that it outputs 0 with probability 1/2 and 1 with probability 1/2.

b. Analyze the expected running time of your algorithm. Note that while the algorithm itself does not use p, the expected running time should be expressed in terms of p.