CPSC 320 Sample Midterm 2
November 2013

Name: _____ Student ID: _____

Signature: _____

– You have 50 minutes to write the 5 questions on this examination. A total of 40 marks are available.

– **Justify all of your answers.**

– You are allowed to bring in one hand-written, double-sided 8.5 x 11in sheet of notes, and nothing else.

– Keep your answers short. If you run out of space for a question, you have written too much.

– The number in square brackets to the left of the question number indicates the number of marks allocated for that question. Use these to help you determine how much time you should spend on each question.

– Use the back of the pages for your rough work.

– **Good luck!**

Question	Marks
1	
2	
3	
4	
5	
Total	

UNIVERSITY REGULATIONS:

– Each candidate should be prepared to produce, upon request, his/her UBC card.

– No candidate shall be permitted to enter the examination room after the expiration of one half hour, or to leave during the first half hour of the examination.

– CAUTION: candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.

1. Having at the place of writing, or making use of, any books, papers or memoranda, electronic equipment, or other memory aid or communication devices, other than those authorised by the examiners.
2. Speaking or communicating with other candidates.
3. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.

– Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.

[9] 1. Sorting and Order Statistics

[3] a. Why is *RandomizedQuicksort* (Quicksort using a random pivot) better than the regular (non-randomized) version of Quicksort?

[6] b. In assignment 6, you wrote an algorithm to find the i^{th} order statistics of a set stored in a binary search tree. Recall that we used an additional piece of data stored in each node N of the tree: the size of the subtree rooted at N. Describe an algorithm that determines, given such a binary search tree T and a key k, the *rank* of k in the set stored in T (that is, if k is the 17^{th} smallest element of the set, then your algorithm should return 17).

[10] 2. Amortized Analysis

[3] a. Can amortized analysis help us derive a better upper-bound on the worst-case running time of a single operation on a data structure? Why or why not?

[3] b. You are using the potential method to obtain a tight upper bound on the running time of a sequence of operations on a data structure. The *brziptle* operation on this (bizarre) data structure executes a constant number of constant time steps, as well as one loop. This loop iterates j times, where the value of j varies from one *brziptle* call to the next, and might be as large as $n-1$ (where n is the number of elements in the data structure). Each loop iteration runs in $\Theta(1)$ time.

Explain what you would need to do in order to prove that the amortized cost of the *brziptle* operation is in $\Theta(1)$.

[4] c. Consider a data structure that supports the following three operations:

- `insert(x)`, that runs in $\Theta(1)$ time.
- `search(x)`, that runs in $O(j)$ time where j is the number of elements that it will visit before finding x (in the worst case, j could be the number of elements of the data structure).
- `delete(x)`, that first calls `search(x)` and then performs a $\Theta(1)$ time deletion.

Assume furthermore that elements "wear out": that is, the fourth time that an elements is visited during a `search` operation, it is deleted (in $\Theta(1)$ time).

Suggest a potential function that could be used to prove that the amortized cost of every operation on this data structure is in $\Theta(1)$ (you do **not** need to compute the amortized costs). Justify your answer briefly.

[4] 3. Explain the best way to find a tight upper bound on the solution of the recurrence relation

$$T(n) = \begin{cases} 4T(n/2+3) + 5n^2 \log n & \text{if } n \geq 8 \\ \Theta(1) & \text{if } n < 7 \end{cases}$$

and state what you believe the upper bound to be (you do not need to prove your upper bound).

[8] 4. Prove an upper bound on the function $T(n)$ defined by

$$T(n) = \begin{cases} 2T(n/2) + T(2n/3) + 2T(n/6) + n^2 & \text{if } n \geq 6 \\ 1 & \text{if } n \leq 5 \end{cases}$$

Your grade will depend on the quality of the bound you provide (that is, showing that $T(n) \in O(100^n)$, while true, will not give you many marks).

[9] 5. Design a divide-and-conquer algorithm that takes as input an array of integers, and returns both the minimum and the maximum elements of the array. Your algorithm should make at most $3n/2 + c$ comparisons where c is some small (additive) constant.