CPSC 223: Final-Exam Instructions

The final exam grade for CPSC 223 will be split into two parts consisting of a written report (50 points) and an in-class exam (75 points). Be sure to look at the class webpage for the day and time of the in-class exam. Instructions for the written report are provided below. The written report is due by the end of the day on *Wednesday, May 4th.*

Written Report Instructions:

You must write a short report on the different Map implementations we worked on throughout the semester. You can think of this as the final "lab report" for the assignments. The implementations to consider include using a resizable array (`ArrayMap`), linked list (`LinkedMap`), sorted array list with binary search (`BinSearchMap`), hash table (`HashMap`), (non-balanced) binary search tree (`BSTMap`), and balanced binary search tree (`AVLMap`). Your report will be graded on the following, ordered by importance.

- *Completeness.* See below for details.
- *Correctness.* No obvious errors in data, summaries, or analysis.
- *Analysis Quality.* Correctness and depth of analysis throughout.
- *Clarity and Organization.* Note there is no minimal, target, or maximum length for your report—it should be "as short as possible, but not any shorter."

Your report must include the following two sections and be submitted via GitHub classroom as a single PDF file by the due date.

(1) Comparison of Performance and Analytical Results. The goal of this section is to summarize the results of each of the key-value pair map implementations according to the five main operations (insert, erase, contains, find keys, and sorted keys). Your comparison should include both Big-O (asymptotic) complexity and runtime performance results. Your comparison should focus on the main differences and the trade-offs among the different approaches (with respect to the five main operations). This part of the report is meant to summarize and clarify the data structures we examined and their performance advantages and disadvantages. As part of your comparison, you must include and fill in the following table (with corresponding Big-O complexities). You also must provide performance graphs from previous assignments as part of your narrative (i.e., to support your analysis). In particular, you must weave the graphs into your discussion as opposed to just pasting them in without any discussion.

	`LinkedMap`	`ArrayMap`	`BinSearchMap`	`HashMap`	`BSTMap`	`AVLMap`
Insert						
Erase						
Contains						
Find Keys						
Sorted Keys						

(2) Application Scenarios. In this section, consider the following scenarios regarding the use of our key-value pair map API. For each scenario, recommend a map implementation (of those we studied) that would minimize the overall scenario cost. State the implementation and explain why you think it would be the "best" implementation for the scenario. Your justification should incorporate worst-case order-of-magnitude costs using Big-O notation and the results of the performance evaluation as appropriate. For each scenario, assume a large number of key-value pairs are already "bulk loaded" into the map, and this bulk loading has the same cost for all of the implementations.

(a). An application where the two main operations are insert and contains, which are performed at approximately the same frequency.

(b). An application where the two main operations are insert and find keys, but where the keys need to be returned in sorted order. Assume find keys is performed much more frequently than insert. Since some map implementations don't return the results of a range search in sorted order, an additional step may be needed to do the sort (which could potentially still lead to more efficient overall operation cost).

(c). An application where a frequent number of insert and erase operations are performed such that a series of inserts are done, followed by a series of erases, and so on. Assume that the overall size of the map ends up growing slowly (with slightly more insert than erase calls over time).