Assignment 6

COMPSCI 2CO3: Data Structures and Algorithms–Fall 2025

Deadline: November 14, 2025

Department of Computing and Software
McMaster University

Please read the *Course Outline* for the general policies related to assignments.

Plagiarism is a *serious academic offense* and will be handled accordingly.
All suspicions will be reported to the *Office of Academic Integrity*
(in accordance with the Academic Integrity Policy).

This assignment is an *individual* assignment: do not submit work of others. All parts of your submission *must* be your own work and be based on your own ideas and conclusions. Only *discuss or share* any parts of your submissions with your TA or instructor. You are *responsible for protecting* your work: you are strongly advised to password-protect and lock your electronic devices (e.g., laptop) and to not share your logins with partners or friends!

If you *submit* work, then you are certifying that you are aware of the *Plagiarism and Academic Dishonesty* policy of this course outlined in this section, that you are aware of the Academic Integrity Policy, and that you have completed the submitted work entirely yourself. Furthermore, by submitting work, you agree to automated and manual plagiarism checking of all submitted work.

Late submission policy. We provide a general *shared grace period* for all assignments of 5 days (120h) that students can utilize whichever way they seem fit without further notice or approval. For example, a student can hand in *one* assignment 100h late, or hand in 4 assignments 20h late (as long as the total stays below 120h). Beyond the above grace period, we require students to *submit on time*. Late submissions are not accepted and will result in a grade of zero. The above grace period *does not stack* with MSAFs or SAS accommodations.

Problem 1. Consider the sequence of values $S = [78, 6, 88, 19, 77, 79, 94, 7, 1, 36, 934]$.

P1.1. Draw the *left-leaning* red-black tree obtained by adding the values in S in sequence. Show each step.

P1.2. Consider the hash function $h(x) = (x + 5) \bmod 12$ a hash-table of 12 table entries that uses hashing with separate chaining. Draw the hash-table obtained by adding the values in S in sequence. Show each step.

P1.3. Consider the hash function $h(x) = (x + 5) \bmod 12$ a hash-table of 12 table entries that uses hashing with linear probing. Draw the hash-table obtained by adding the values in S in sequence. Show each step.

Do *not* spend time drawing beautiful trees or tables: a clear textual representation is good enough.

Problem 2. Consider a list L of pairs. A common operation on such a list is *aggregating* the list on a given field and then counting, per value of that field, the number of tuples that have that value. For example, consider a list of pairs that holds course enrollment information (pairs *student identifier* and *course code*). If we aggregate this list on *student identifier*, then we compute, per *student*, the number of courses that student takes. Likewise, if we aggregate this list on *course code*, then we compute, per *course*, the number of students enrolled in that course.

P2.1. Write an algorithm that, given a set of pairs L, computes the aggregate

$$\mathrm{aggr}(L) = \{(a, \mathrm{count}(a, L)) \mid \mathrm{count}(a, L) \neq 0\} \text{ with } \mathrm{count}(a, L) = |\{b \mid (a, b) \in L\}|.$$

The complexity of your algorithm must be expected $O(|L|)$, your algorithm cannot change the list L, and your algorithm can use at-most $O(|\mathit{result}|)$ memory with $|\mathit{result}|$ the size of the result (the number of pairs $(a, \mathrm{count}(a, L)) \in \mathrm{aggr}(L)$).

You can only assume that you can efficiently *iterate* over all values in L. You *cannot* make any further assumptions on how list L is structured (e.g., you cannot assume that L is sorted).

Hint: Feel free to first solve this problem when you know that $|\mathit{result}| = k$ beforehand. Next, argue why you do not need this assumption in practice.

P2.2. Argue why your algorithm correctly computes $\mathrm{aggr}(L)$ and argue why your algorithm meets the complexity requirements stated in Problem P2.1.

Consider two sets-of-pairs F and S. For example, set F can hold student information (*student identifier* and *name*) and list S can hold course enrollment information (*student identifier* and *course identifier*). When processing data, a typical operation on F and S is computing the *semi-join*:

$$\mathrm{SemiJoin}(F, S, \delta) = \{(a, b) \mid (a, b) \in F \wedge (a, \delta) \in S\}.$$

e.g., if F holds student information, S holds course enrollment information, and $\delta =$ '*COMPSCI 2C03*', then this semi-join will return the student information of all students that take the course *COMPSCI 2C03*.

P2.3. Write an algorithm that, given sets-of-pairs F and S and value δ, *correctly computes* the above semi-join $\mathrm{SemiJoin}(F, S, \delta)$. The complexity of your algorithm must be expected $O(|F| + |S|)$, your algorithm cannot change the lists F and S, and your algorithm can use at-most $O(|S|)$ extra memory besides the memory used to store the result.

You can only assume that you can efficiently *iterate* over all values in F and S. You *cannot* make any further assumptions on how F and S are structured (e.g., you cannot assume that they are sorted).

Hint: Feel free to first solve this problem when you know that $|\mathit{result}| = k$ beforehand. Next, argue why you do not need this assumption in practice.

P2.4. Argue why your algorithm correctly computes $\mathrm{SemiJoin}(F, S, \delta)$ and argue why your algorithm meets the complexity requirements stated in Problem P2.3.

Assignment Details

Write a report in which you solve each of the above problems. Your submission:

1. must start with your name, student number, and MacID;
2. must be a PDF file;
3. must have clearly labeled solutions to each of the stated problems;
4. must be clearly presented;
5. must *not* be hand-written: prepare your report in LaTeX or in a word processor such as Microsoft Word (that can print or export to PDF).

Submissions that do not follow the above requirements will get a grade of zero.

Grading

Each problem counts equally toward the final grade of this assignment.