Assignment 5

COMPSCI 2CO3: Data Structures and Algorithms–Fall 2025

Deadline: November 7, 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.

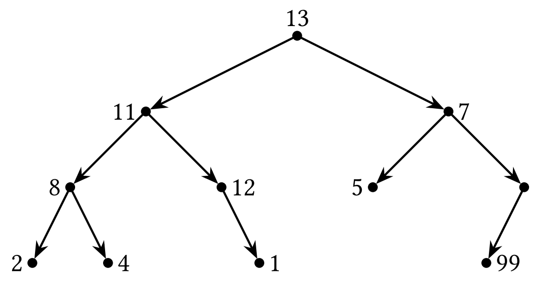
Problem 1. Consider the sequence of values $S = [78, 6, 88, 19, 77, 79, 94, 7, 1, 36, 934]$.

P1.1. Draw the min heap (as a tree) obtained by adding the values in S in sequence. Show each step.

P1.2. Draw the max heap (as a tree) obtained by adding the values in S in sequence. Show each step.

P1.3. Draw the binary search tree obtained by adding the values in S in sequence. Show each step.

Do *not* spend time drawing beautiful trees: a clear textual representation is good enough. Consider, for example, the following tree (note this tree is neither a heap or a binary search tree):



We can represent this tree textually via the very compact representation:

```
13 (11 (8 (2 4)
        12 (* 1))
     7 (5
        6 (99 *))).
```

In this notation, we used * as a place holder for a missing child for those nodes that only have a single child.

Problem 2. Consider a list E that holds information on events: the list E contains pairs (*start-date*, *end-date*) that denote the start and end date of events. To make operations on this list easier, the list E is already sorted on lexicographical order (hence, pairs in E are sorted on increasing start-date and, for events with equal start-date, on increasing end-dates). We are interested in finding events held during a given period of time.

P2.1. Consider the range query

$$\text{Range}(E, a, b) = \{(s, e) \in E \mid a \leq s \leq b\}$$

that returns all events that start in the period of time starting at a and ending at b. Write an algorithm that, given list-of-events E and time points a and b, *correctly computes* the range query $\text{Range}(E, a, b)$. The complexity of your algorithm must be worst-case $O(\log(|E|) + \textit{result}))$ with $|\textit{result}|$ the size of the result (the number of pairs $(s, e) \in \text{Range}(E, a, b)$).

P2.2. The range query $\text{Range}(E, a, b)$ does *not* return all events that are held during the period of time starting at a and ending at b: an event $(s, e) \in E$ can *start before* a and *not end before* a ($s < a \leq e$). Unfortunately, it is hard to find these events without inspecting all events in E that start before a.

Fortunately, we can build an *index structure* inspired by a search tree to easily find such values. Provide the design of such a search tree $T(E)$ that can easily return all events that start before a. Hence, your tree design should support the operation

$$\text{StillActive}(T(E), a) = \{(s, e) \in E \mid s < a \leq e\}$$

that returns all events in E that are *still active* during a. What are the *node values* used to navigate your tree $T(E)$? What does it mean to navigate to a left child or a right child in your tree $T(E)$? What other information do nodes of your tree $T(E)$ hold? How do you answer $\text{StillActive}(T(E), a)$?

Assuming you have a pre-constructed tree $T(e)$, the complexity of answering $\text{StillActive}(T(E), a)$ must be worst-case $O(\log(|E|) + \textit{result})$ with $|\textit{result}|$ the size of the result (the number of events $(s, e) \in \text{StillActive}(T(E), a)$. Your tree can use at-most $O(|E|)$ memory.

P2.3. Given a list E sorted on lexicographical order. Provide an algorithm that constructs $T(e)$ in worst-case $O(|E|)$.

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.