# Assignment 2

## SFWRENG 2CO3: Data Structures and Algorithms–Winter 2024

Deadline: January 28, 2024

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.* Late submissions will receive a late penalty of 20% on the score per day late (with a five hour grace period on the first day, e.g., to deal with technical issues) and submissions five days (or more) past the due date are not accepted. In case of technical issues while submitting, contact the instructor *before* the deadline.

**Problem 1.**

P1.1. Consider an initially-empty stack $S$ and the sequence of operations

$$\textsc{push}(S, 3), \textsc{pop}(S), \textsc{push}(S, 17), \textsc{push}(S, 5), \textsc{push}(S, 15), \textsc{pop}(S), \textsc{pop}(S), \textsc{pop}(S).$$

Illustrate the result of each operation (clearly indicate the content of the stack after the operation and, in case of a POP, the value returned by the operation).

P1.2. Consider an initially-empty queue $Q$ and the sequence of operations

$$\textsc{enqueue}(S, 3), \textsc{dequeue}(S), \textsc{enqueue}(S, 17), \textsc{enqueue}(S, 5), \textsc{enqueue}(S, 15), \textsc{dequeue}(S), \textsc{dequeue}(S), \textsc{dequeue}(S).$$

Illustrate the result of each operation (clearly indicate the content of the queue after the operation and, in case of a DEQUEUE, the value returned by the operation).

P1.3. Assume we have a stack implementation `MyDynArrayStack` using dynamic arrays: the implementation supports $N$ PUSH operations with an amortized runtime complexity of $\Theta(1)$ per operation, and the implementation supports POP, EMPTY, and SIZE operations with a runtime complexity of $\Theta(1)$.

Provide a *queue* implementation that uses `MyDynArrayStack` and supports any valid sequence of $N$ ENQUEUE and DEQUEUE operations with an amortized runtime complexity of $\Theta(1)$ per operation. Explain why your implementation has the stated amortized runtime complexity for ENQUEUE and DEQUEUE.

**Problem 2.** Consider the relations courses(*prog*, *code*, *name*) (that models courses named *name* and identified by the program *prog* the course is part of, e.g., SFWRENG, and the course code *code*, e.g., 2C03) and enrolled(*prog*, *code*, *sid*) (that models students with identifier *sid* enrolling for a course identified by program *prog* and course code *code*).

We want to compute the list of all pairs (*sid*, *name*) in which *sid* is a student identifier *s* and *name* is the name of a course the student with identifier *s* is enrolled in. To compute this list, we developed the following two *nested-loop algorithms*:

**Algorithm CEJoin**(courses, enrolled) **:**

```
1: output := ∅.
2: for (p_c, c_c, n_c) ∈ courses do
3:     for (p_e, c_e, s_e) ∈ enrolled do
4:         if p_c = p_e and c_c = c_e then
5:             add (s_e, n_c) to output.
6:         end if
7:     end for
8: end for
```

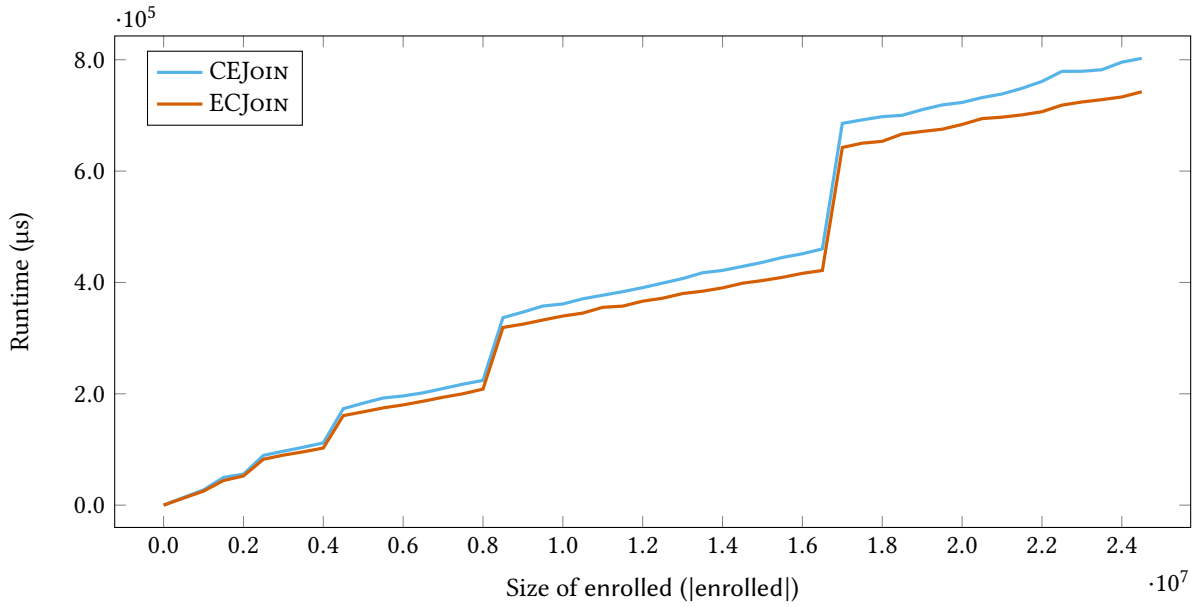
**Algorithm ECJoin**(courses, enrolled) **:**

```
1: output := ∅.
2: for (p_e, c_e, s_e) ∈ enrolled do
3:     for (p_c, c_c, n_c) ∈ courses do
4:         if p_c = p_e and c_c = c_e then
5:             add (s_e, n_c) to output.
6:         end if
7:     end for
8: end for
```

Assume we have significantly more students enrolled for courses than courses (|enrolled| > |courses|).

P2.1. Assume we are running algorithms CEJoin and ECJoin on a computer $\mathbb{C}$ where *every* instruction takes *exactly the same amount of time* to execute. Argue why CEJoin must be faster than ECJoin when run on the computer $\mathbb{C}$.

P2.2. A student decided to implement both algorithms in C++ (see `impl_22.cpp` for the implementation) and found that ECJoin is faster than CEJoin (see `data_22.tsv` for the measurement data):




On real-world systems, one would always expect this outcome. Explain why this is the case.

**Hint:** You may assume that the processor uses fast *caches* when reading memory: if the last $M$ values my program reads from memory (for some small $M$) are $v_1, \ldots, v_M$, then reading any value $v \in \{v_1, \ldots, v_M\}$ again will typically be significantly faster than reading other values.

P2.3. The measurements in the above figure have a few sudden *jumps*, e.g., at 1 500 000, 2 500 000, 4 500 000, 8 500 000, and 17 000 000. Explain what causes these jumps.

P2.4. Write an algorithm that efficiently computes the same result as CEJoin and ECJoin in all-case $\Theta(|\text{enrolled}| \log_2(|\text{courses}|))$. In the design of your algorithm, you may require that either enrolled or courses is ordered. Argue why your algorithm is correct and why it has the runtime complexity we specified.

## 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 exported 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.