Assignment 2

COMPSCI 2CO3: Data Structures and Algorithms–Fall 2025

Deadline: October 3, 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. 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

$$\textsc{push}(S,7),\ \textsc{push}(S,9),\ \textsc{pop}(S),\ \textsc{push}(S,12),\ \textsc{push}(S,13),\ \textsc{pop}(S),\ \textsc{pop}(S),\ \textsc{pop}(S).$$

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

$$\textsc{enqueue}(Q,7),\ \textsc{enqueue}(Q,9),\ \textsc{dequeue}(Q),\ \textsc{enqueue}(Q,12),\ \textsc{enqueue}(Q,13),\ \textsc{dequeue}(Q),\ \textsc{dequeue}(Q),\ \textsc{dequeue}(Q).$$

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. For a memory-constrained application, we need two high-performance stacks that will hold at-most n values in total (for example, if $n = 13$ and the first stack holds 5 values, then the second stack can hold up-to $13 - 5 = 8$ values).

Show how to implement these two stacks using a single array holding n distinct values. Your solution must be able to push values to and pop values from each stack in worst-case $O(1)$.

Problem 2. Assume $n \geq 0$. The *square root* of n is the number m such that $m^2 = n$. Next, we will develop an efficient algorithm to compute the square root of n.

P2.1. Assume the square root is an integer. Consider the following algorithm.

Algorithm ComputeSquareRoot(n) **:**

```
1: for i := 0 to ⌈n/2⌉ (inclusive) do
2:     if i^2 = n then
3:         return i.
4:     end if
5: end for
```

Is this algorithm correct? If so: provide an invariant and prove that the algorithm is correct. Otherwise, provide details on how to fix the algorithm.

P2.2. What is the worst-case complexity of this algorithm?

P2.3. Assume the square root is an integer. Provide an alternative to the above algorithm that computes the square root of n in $O(\log_2(n))$. Argue why your algorithm is correct and has the stated complexity.

P2.4. Now assume the square root can be a non-integer positive number. Show how to improve your algorithm for Problem P2.3 to find an *exact approximation* of the square root of n: given n and a factor $\epsilon < 1$, your algorithm must find a value m such that $n - \epsilon \leq m^2 \leq n + \epsilon$. Argue why your algorithm is correct and provide the complexity of your algorithm.

You may assume that any number can be stored with arbitrary precision and that any elementary math operation (addition, subtraction, multiplication, and division) on numbers can be performed in constant time.

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.