University of Waterloo
CS240E, Spring 2025
Written Assignment 1

Due Date: Tuesday, May 20, 2025 at 5pm, with grace period until 7:59pm

Be sure to read the assignment guidelines (`https://student.cs.uwaterloo.ca/~cs240e/s25/assignments.phtml#guidelines`). Submit your solutions electronically to Crowdmark. Ensure you have read, signed, and submitted the Academic Integrity Declaration AID01.

Grace period: submissions made before 8pm on May 20 will be accepted without penalty. Please note that submissions made after 8pm **will not be graded** and may only be reviewed for feedback.

Question 1 [9 marks]

There are two different definitions of 'little-omega' in the literature (to distinguish them, we will call them ω_0 and ω_1 here). Fix two functions $f(x), g(x)$. We say that

- $f(x) \in \omega_0(g(x))$ if for all $c > 0$ there exists $n_0 > 0$ such that $|f(x)| \geq c|g(x)|$ for all $x \geq n_0$, and
- $f(x) \in \omega_1(g(x))$ if $g(x) \in o(f(x))$.

Show that these two definitions are equivalent, i.e., $f(x) \in \omega_0(g(x))$ if and only if $f(x) \in \omega_1(g(x))$. Your proof must be from first principle, i.e., directly using the definitions (do not use the limit-rule). Note that $f(x), g(x)$ are not necessarily positive.

Question 2 [3+6=9 marks]

Motvation: In class we often use a sloppy recursion, or assume that "n is divisible as needed". The following question illustrates that, with some limitations, this approach is justified.

a) Show that the following statement is **true**.

"Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a monotonically increasing function. Assume that $f(x) \leq x$ whenever x is a power of 2, i.e., $x = 2^k$ for some integer $k \geq 0$. Then $f(x) \in O(x)$."

b) Show that the following statement is **false**.

"Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a monotonically increasing function. Assume that $f(x) \leq x$ for infinitely many integers, i.e., for any N there exists an integer $x \geq N$ with $f(x) \leq x$. Then $f(x) \in O(x)$."

Reminder: To show that a statement is false, you need to give an example that satisfies all assumptions of the statement, but does not satisfy the conclusion.

Question 3 [9 marks]

Consider a (max-oriented) meldable heap H that holds n integers. Describe an algorithm that is given H and an integer x, and that finds all items in H for which the priority is at least x. (Note that x may or may not be in H.) Your algorithm should have $O(1+s)$ worst-case run-time, where s is the number of items that were found.

Question 4 [2+6+4=12 marks]

To reduce the height of the heap one could use a d-way heap. This is a tree where each node contains up to d children, all except the bottommost level are completely filled, and the bottommost level is filled from the left. It also satisfies that the key at a parent is no smaller than the keys at all its children.
Your answers must work for arbitrary d, even if d is not constant.

a) Explain how to store a d-way heap in an array A of size $O(n)$ such that the root is at $A[0]$. Also state how you find parents and children of the node stored at $A[i]$. You need not justify your answer.

b) What is the height of a d-ary heap on n nodes? Give a tight asymptotic bound that depends on d and n. You may assume that n and d are sufficiently big (e.g. $d \geq 3$ and $n \geq 10$). Note that d is not necessarily a constant.

c) Assume that $n \geq 4$ is a perfect square. What is the height of a d-ary heap for $d = \sqrt{n}$? Give an exact bound (i.e., not asymptotic).

Question 5 [3+7+3=13 marks]

How would you implement *increase-key*(z,k) in a binomial heap? The method is given as parameter a node z and a key k and it should increase the key of z to k if it was smaller before.

a) Prof. B. Fuddled thinks that they can implement this using *fix-up* as follows:

Algorithm 1: *increase-key*(z,k)

```
if (k > z.key()) then
    z.key ← k
    while p ← z.parent is not NULL and
      p.key < z.key do            // do fix-up
        swap key-value pairs of z and p
        z ← p
```

Show that Prof. Fuddled is incorrect. Thus, give an example of a flagged tree that satisfies the binomial-heap-order property, indicate a node z and a key $k > z.key$, and show that calling *increase-key(z,k)* results in a flagged tree that does not satisfy the binomial-heap-order property. (Try to keep your tree small, no more than 16 nodes.)

b) Give a method to implement *increase-key* in a flagged tree with the binomial-heap-order property, with worst-case run-time $O(\log n)$.

c) Recall that *decrease-key*(z, k) is given a node z and a key k and should decrease the key of z to k if it was bigger before. Show that this operation can be reduced to the other operations. Specifically, show that if a priority queue realization supports *size*, *find-max*, *delete-max*, *increase-key* and *insert* with $O(f(n))$ run-time, then you can also realize *decrease-key* with $O(f(n))$ run-time.