# University of Waterloo
# CS240E, Winter 2023
# Midterm Post-Mortem

## 1 Fill in missing ($11 \times 1 = 11$ marks)

- For part g: a very common error were answers that began with 3.
- For part i: many students gave the answer of 5. The answer to this question is that a rebuild is impossible regardless of the inserted key (the problem statement specifies that one question has the answer of "this could not have happened").

## 2 Short-answer ($2 + 3 + 3 + 4 = 10$ marks)

- This question was generally done well.
- Many solutions skipped Q2(d). We will discuss this problem in tutorial.

## 3 Algorithm analysis ($2 + 2 + 2 + 3 + 3 + 3 = 15$ marks)

- Part d: many solutions mistakenly resolved the recurrence to $\Theta(n \log n)$ rather than $\Theta(n)$.
- Many solutions only showed the upper big-O bound, while a tight big-$\Theta$ bound was necessary.

## 4 Sorting ($7 + 7 = 14$ marks)

- Part a: the common error was sorting the array $D$ before computing the array $S - D$
- Part b: the common error was not handling the case where several items were decreased in a row. Several solutions incorrectly assumed that the array $T$ is part of the input.

## 5 Skip lists ($3 + 3 = 6$ marks)

- For part $a$, A common error was manipulating the stack from *getPredecessors*, rather than traversing $S_0$.
- Also, explicitly finding the predecessors of $a$ or $b$ is not necessary, as we implemented *skipList::search* in lecture.

- For part $b$, solutions were well-done. Several submissions checked for whether the next node exists (rather than checking if it contains the key $\infty$).
- In this problem, it suffices to state the expected height of a tower is constant because we did it in class. Many solutions gave the details.

## 6 Randomization ($3 + 4 = 7$ marks)

- For part a, very few solutions gave a specific counter-example.
- Part b was generally done well. Several solutions had hard-to-read pseudocode.

## 7 Amortized analysis ($8$ marks)

- Many solutions forgot time units or did not define them explicitly
- Many solutions did not separate rebuild from delete
- Some solutions proved the actual runtimes from scratch (which was not needed)

## 8 Building binomial heaps ($5$ marks)

- Many solutions split the input into blocks of sizes determined from the binary representation of $n$, and built flagged trees on those blocks. A commmon error there was not ensuring the heap-order property. One way to do that is to find the maximum element (to use as root), and heapify the rest (all in linear time).