CSC 263H5S 2020 Midterm Test
Duration — 80 minutes
Aids allowed: one double-sided 8.5"x11" sheet

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Last Name: _____ **First Name:** _____

☐ Registered Lecture Section: L0101/L0102 (Sushant Sachdeva)
☐ Registered Lecture Section: L0103 (Jessica Burgner-Kahrs)

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above, **write your name on the back of the test**, and read the instructions below.)

This test consists of 5 questions on 14 pages (including this page). When you receive the signal to start, please make sure that your copy is complete. If you write your answer in the space for rough work, indicate clearly what you want marked.

1: _____/12
2: _____/ 9
3: _____/ 6
4: _____/15
5: _____/12

TOTAL: _____/54

Question 1. [12 MARKS]

Part (a) [2 MARKS]

Consider whether the array below holds a heap:

85	22	41	5	21	13	28	15		

If it holds a heap, What type of a heap is it? (Mark the box next to the correct answer with $\boxed{\surd}$.)

- ☐ the array is a MIN heap
- ☐ the array is a MAX heap
- ☐ the array is NOT a valid heap

Part (b) [3 MARKS]

Sort the following three functions in ascending order according to their asymptotic growth.

$$f_1(n) = \frac{n^2}{\log n^{2/3}} \quad f_2(n) = n \log n^{3/2} \quad f_3(n) = \frac{3}{2} n^{3/2}$$

Write $f_i = \Theta(\ldots)$ in ascending order:

_____ < _____ < _____

Part (c) [3 MARKS]

Consider the following array.

16	3	17	5	42	25	31	39

When running `RandomizedQuicksort` on the above array, what is the probability that `17` and `25` will be compared. Justify your answer in one sentence.

Part (d) [4 MARKS]

List the following algorithms and data structure operations according to their asymptotic **worst-case** runtime in the table provided below.

- `quickSort()` in its best implementation on an array of length n
- `last()` on a doubly-linked list of length n
- `insert()` on a hash table with open addressing using linear probing
- `search()` on a balanced binary search tree with n nodes
- `delete()` on a binary search tree with n nodes
- `extractMax()` on a heap with n elements
- `heapSort()` on an array with n elements
- `insertionSort()` on an array with n elements

$\Theta(1)$	
$\Theta(\log n)$	
$\Theta(n)$	
$\Theta(n \log n)$	
$\Theta(n^2)$	

Question 2. [9 MARKS]

Part (a) [2 MARKS]

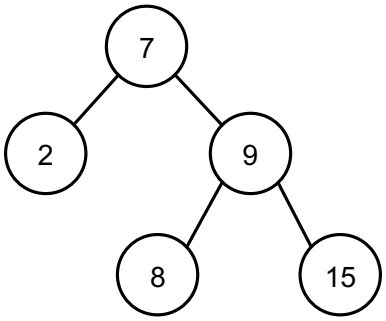
The root of a heap is at level 0. How many nodes are present in the k-*th* level assuming the heap is a complete binary tree?

Nodes in the k-*th* level: _____

Number of nodes in all levels above the k-*th*: _____

Part (b) [3 MARKS]

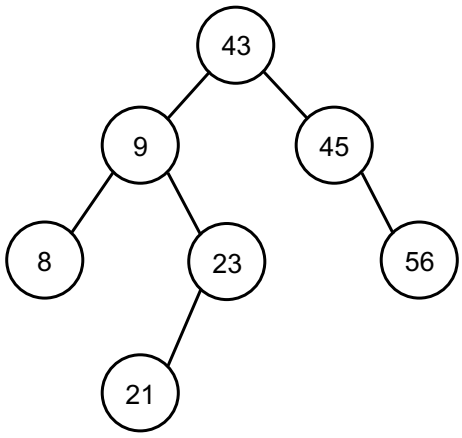
Consider the following AVL tree.



Insert a new node 17 into the tree. Draw the tree after the insert operation (before any rotation) and after each subsequent rotation operation. Specify which type of rotation operation you used in each step.

Part (c) [4 MARKS]

Consider the following AVL tree.



Delete node 45 from the tree. Draw the tree after the delete operation (before any rotations) and after each subsequent rotation operation. Specify which type of rotation operation you used in each step.

Question 3. [6 MARKS]

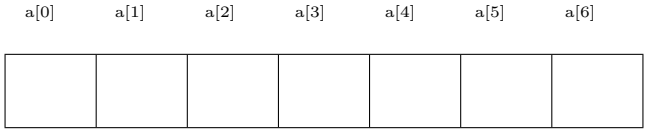
Part (a) [3 MARKS]

Insert the following sequence of values into the array **a** of length 7 below using the hash function

$$h(k, i) = ((k \mod 7) + i) \mod 7)$$

derived using the division method with linear probing (with $i = 0, 1, \ldots$).

Sequence: 5, 7, 4, 11, 0, 12



Part (b) [3 MARKS]

Name and briefly explain three characteristics to qualitatively assess how good a hash function is.

Question 4. [15 MARKS]

We can implement a queue Q using two stacks H and T as follows: Think of the stack T as containing the "tail" of the queue (i.e., the recently inserted items), with the most recently inserted item at the top. Think of the stack H as containing the "head" of the queue (i.e., the older items), with the oldest item of the queue at the top. Conceptually, Q consists of T and H placed "back-to-back".

To EnQueue an item x on Q, we actually Push x into T. To DeQueue an item, we Pop H, provided H is not empty. If H is empty, we transfer the items of T to H (by popping each item of T and then pushing it into H), and then Pop H.

These algorithms are given below in pseudo-code. (We assume that initially the stacks H and T are empty, and that the function StackEmpty(T) returns **true** if T is an empty stack and **false** otherwise.)

```
EnQueue(Q, x)
    Push(T, x)
```

```
DeQueue(Q)
    if StackEmpty(H) then
        loop
            exit when StackEmpty(T)
            x := Pop(T)
            Push(H, x)
        end loop
    end if
    return Pop(H)
```

Assume that each Push, Pop and StackEmpty operation takes $\Theta(1)$ time.

Part (a) [6 MARKS]

Consider performing a sequence of m operations, consisting of EnQueue and DeQueue, starting from empty H and T.

1. In such a sequence, what is the worst-case time complexity of a single EnQueue operation?
2. In such a sequence, what is the worst-case time complexity of a single DeQueue operation?

Obtain **tight** $\Theta(\cdot)$ complexity guarantees by deriving matching upper ($O(\cdot)$) and lower ($\Omega(\cdot)$) bounds.

Part (b) [9 MARKS]

Use the accounting method to prove that the amortized time complexity of each operation in a sequence of m ENQUEUE and DEQUEUE operations is $O(1)$.

Carefully and precisely state:

1. The amount of credit (dollars) each ENQUEUE and DEQUEUE operation must start with (before they are executed)
2. Which credits (dollars) do you spend to cover for the time taken by an ENQUEUE operation? Justify your answer by showing that you always have the required credits available (in other words, you don't go broke after the operation)
3. Which credits (dollars) do you spend to cover for the time taken by a DEQUEUE operation? Justify your answer by showing that you always have the required credits available (in other words, you never go broke after the operation)

For the above answers, consider justifying your answer by stating the invariant maintained for the credits (dollars) remaining at the end of each operation. In other words, by describing, how much credit (money) is remaining after **every** operation.

Assume that the only operations that you have to account for in the code are PUSH, POP, and STACK-EMPTY, where each of them takes one unit of time.

Question 5. [12 MARKS]

In class, we described the `MaxPriorityQueue` ADT.

Consider a new ADT `BetterMaxPriorityQueue` that in addition to the standard operations on a `MaxPriorityQueue`, also supports the operation `Delete(Q, x)`, which, given a pointer to a node `x` that is already in `BetterMaxPriorityQueue` `Q`, removes `x` from `Q`.

Describe a data-structure that implements the ADT `BetterMaxPriorityQueue` such that the operations `Enqueue(Q,x)`, `Dequeue(Q)`, and `Delete(Q, x)` have a worst-case time complexity of $O(\log n)$, where n is the number of elements in `Q`, and `PeekFront(Q)` has a worst-case time complexity of $O(1)$.

Give a detailed description of your data-structure, and explain why it meets all of the above complexity requirements.

As usual, please do not repeat algorithm details or runtime analyses from class or the textbook just directly refer to known results as needed.

[This page will be marked only if you clearly indicate the part that should be marked.]

[This page will be marked only if you clearly indicate the part that should be marked.]

Last Name: _____ **First Name:** _____

[This page will be marked only if you clearly indicate the part that should be marked.]