Duration: **110 minutes**
Aids Allowed: **One *single*-sided *hand*written *8.5"×11"* aid sheet.**

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

Last (Family) Name(s): _____

First (Given) Name(s): _____

Lecture Section: ☐ L0101 (M. Craig) ☐ L0201 (M. Craig) ☐ L0301 (L. Zhang) ☐ L5101 (F. Pitt)

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This term test consists of 8 questions on 16 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete, fill in the identification section above, and write your name on the back of the last page.*

Answer each question directly on the test paper, in the space provided, and use one of the "blank" pages for rough work. If you need more space for one of your solutions, use a "blank" page and *indicate clearly the part of your work that should be marked.*

In your answers, you may use without proof any theorem or result covered in lectures, tutorials, problem sets, assignments, or the textbook, as long as you give a clear statement of the result(s)/theorem(s) you are using. You must justify all other facts required for your solutions.

Write up your solutions carefully! In particular, use notation and terminology correctly and explain what you are trying to do —part marks *will* be given for showing that you know the general structure of an answer, even if your solution is incomplete.

If you are unable to answer a question (or part of a question), remember that you will get 20% of the marks for any solution that you leave *entirely blank* (or where you cross off everything you wrote to make it clear that it should not be marked).

MARKING GUIDE

Nº 1:	_____/ 5
Nº 2:	_____/ 6
Nº 3:	_____/ 4
Nº 4:	_____/ 4
Nº 5:	_____/ 5
Nº 6:	_____/14
Nº 7:	_____/10
Nº 8:	_____/12
TOTAL:	_____/60

Good Luck!

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 1. [5 MARKS]

Part (a) [1 MARK]

Alice claims the array below holds a heap.

55	22	4	5	20	3	2	1		

What type of heap is this? (Write an "×" in the box next to the correct answer.)

☐ the array is a MIN heap

☐ the array is a MAX heap

☐ the array is NOT a valid heap

Part (b) [2 MARKS]

If you answered in Part (a) that the array was NOT a valid heap, explain your reasoning here.

If you decided that the array was a valid heap, fill in values in the array below to show the resulting heap after inserting the value 25.

Part (c) [2 MARKS]

Bob claims that he can super-efficiently implement a max-priority queue using an **unsorted doubly-linked list**, together with an extra variable ***max*** that stores a pointer to the maximum element in the linked list. According to Bob, the worst-case running times of the operations would be as follows:

- INSERT: just insert at the head of the linked list in $\mathcal{O}(1)$ time.
- EXTRACTMAX: just delete the node pointed to by *max* in $\mathcal{O}(1)$ time (because the list is doubly-linked).
- MAX: just return the priority of the node pointed to by *max* in $\mathcal{O}(1)$ time.

Point out Bob's mistake. Be concise.

Question 2. [6 MARKS]

Part (a) [2 MARKS]

The load factor α of an open addressing hash table must satisfy $\alpha \leqslant 1$. Explain why in one short sentence.

Part (b) [3 MARKS]

Consider a hash table with $m = 10$ slots using the hash function $h(k) = k \bmod m$ with open addressing (storing every item directly into the table, **without** linked lists). The collision strategy is non-linear probing using the probe sequence $h(k, i) = (k + i^2) \bmod m$ (for $i = 0, 1, \ldots, m - 1$).

Starting from an empty table, insert the following keys in order: $52, 1005, 96, 92, 2$. Write the result directly in the table provided below.

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

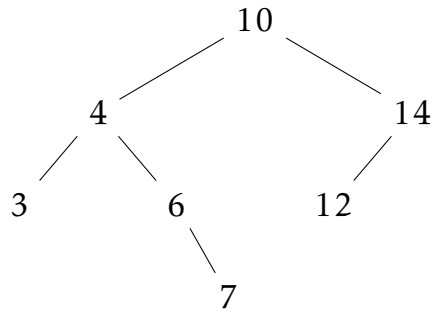
Part (c) [1 MARK]

Give one reason why 10 is a poor choice of m for the hash table in Part (b).

Question 3. [4 MARKS]

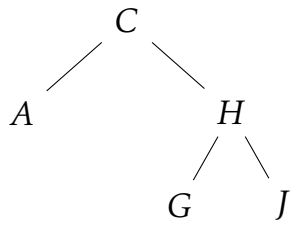
Part (a) [2 MARKS]

Insert a node with key 8 into the AVL tree pictured below. Show the resulting tree by either drawing directly on the original tree or by drawing a new tree in the space on the right.



Part (b) [2 MARKS]

Insert a node with key *E* into the AVL tree pictured below. Show the resulting tree by either drawing directly on the original tree or by drawing a new tree in the extra space.



Question 4. [4 MARKS]

Part (a) [2 MARKS]

Randomized quicksort compares individual pairs of elements but it does not necessarily compare every element to every other element. When the input is the array $[1, 5, 3, 2]$, what is the probability that randomized quicksort compares 1 and 5 directly to each other? Explain your reasoning.

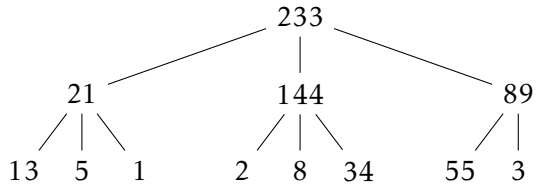
?

Part (b) [2 MARKS]

When the input is the array $[1, 5, 3, 2]$, what is the probability that randomized quicksort compares 1 and 2 directly to each other? Explain your reasoning.

Question 5. [5 MARKS]

In a *ternary* max-heap, every node has exactly three children (except for the leaves and, possibly, one node with fewer than three children). The values stored in each node are ordered according to the same principle as for binary heaps: the value at each node is greater than or equal to the values in the node's children. For example, the following depicts a ternary max-heap.



Just like for binary heaps, we would like to store the heap elements in an array (without creating a linked structure based on Node objects).

Part (a) [2 MARKS]

Draw the array representation for the ternary max-heap above.

Part (b) [3 MARKS]

Given the index i of an element in an array that represents some ternary max-heap, give an *exact expression* for each of the following:

- The index of the root of the heap: ROOT = _____
- The index of the left child of i: LEFT(i) = _____
- The index of the middle child of i: MID(i) = _____
- The index of the right child of i: RIGHT(i) = _____
- The index of the parent of i: PARENT(i) = _____

Question 6. [14 MARKS]

Given an unsorted list of n items where the value of each item is chosen independently from a universe of m values, you want to find the k largest values. Assume that $m \gg n$ (m is much larger than n) and $n > k$. Also, note that the values are **not** necessarily chosen at random.

Analyze each of the approaches below by stating whether it WILL NOT WORK, WORKS BUT MAY BE IMPRACTICAL or is A FINE IDEA. In all cases, explain your thinking. For approaches that will not work, explain why not. For all other approaches, provide the worst-case time complexity and show your work.

Part (a) [2 MARKS]

Insert all n items into an AVL tree using their values as the keys. Starting at the right-most element of the tree, return k of the elements.

☐ WILL NOT WORK ☐ WORKS BUT MAY BE IMPRACTICAL ☐ A FINE IDEA

Explanation/Worst-case time complexity:

Part (b) [2 MARKS]

Create an array of size m where there is a position for each possible value in the universe. Insert all n items into this data structure. Starting with the highest m value and examining each list in decreasing order of m, return items until you have the k items you need.

☐ WILL NOT WORK ☐ WORKS BUT MAY BE IMPRACTICAL ☐ A FINE IDEA

Explanation/Worst-case time complexity:

Part (c) [2 MARKS]

Create an array A of size k. It will represent the k largest values we have seen at each point in time. $A[0]$ will be the largest value. Examine each of the n items in turn comparing it to the elements in A. If it is larger than any one of the elements then insert it into the array and shift the others down dropping the element that was previously at $A[k-1]$.

☐ WILL NOT WORK ☐ WORKS BUT MAY BE IMPRACTICAL ☐ A FINE IDEA

Explanation/Worst-case time complexity:

Question 6. (continued)

Part (d) [2 marks]

Insert all the items into a max-heap. Call ExtractMax k times on the heap.

☐ Will Not Work ☐ Works But May Be Impractical ☐ A Fine Idea

Explanation/Worst-case time complexity:

Part (e) [2 marks]

Insert all the items into a min-heap. Return the **last** k items from the heap.

☐ Will Not Work ☐ Works But May Be Impractical ☐ A Fine Idea

Explanation/Worst-case time complexity:

Part (f) [2 marks]

Sort the items using quick-sort or merge-sort and then return the first k values.

☐ Will Not Work ☐ Works But May Be Impractical ☐ A Fine Idea

Explanation/Worst-case time complexity:

Part (g) [2 marks]

Create a hash-table of size p where p is as large as you can manage given space constraints. Hash all n items to the table. Then remove the largest k items from the table and return them.

☐ Will Not Work ☐ Works But May Be Impractical ☐ A Fine Idea

Explanation/Worst-case time complexity:

Question 7. [10 MARKS]

In this question, you will augment AVL trees to implement the following new operation:

- NUMGREATER(k): return the number of elements with a key strictly greater than k.

Part (a) [2 MARKS]

What additional information will you store at each node of the AVL tree? Be specific.

Part (b) [3 MARKS]

Explain how to maintain your new information during INSERT operations, without affecting the running time by more than a constant factor.

Question 7. (CONTINUED)

Part (c) [5 MARKS]

Give a detailed implementation for operation NUMGREATER. Explain what you are doing (use comments or a brief English description of the main idea of your implementation). Then, analyse the worst case complexity of your implementation.

Question 8. [12 MARKS]

Consider the following algorithm that inserts a value x into a sorted array A. Remember that in this course, the word "array" means a C-style array with a *fixed size* (**not** a Python-style list that grows and shrinks automatically). In this context, assume that A is not entirely full when x is inserted: A contains $A.size$ elements ($A[0], A[1], \ldots, A[A.size-1]$) out of $A.length$ possible locations (the rest of the locations $A[A.size], A[A.size+1], \ldots, A[A.length-1]$ store the special value NIL).

For example, starting with $A = [5, 8, 21, 34, \text{NIL}, \text{NIL}]$, $x = 13$ and calling SORTEDINSERT(A, x) changes the contents of A to $[5, 8, 13, 21, 34, \text{NIL}]$.

```
SortedInsert(A, x):
    # Precondition: A[0] ⩽ ... ⩽ A[A.size − 1] and A[A.size] = ··· = A[A.length − 1] = NIL.
    # First, store x in the first available location.
    k ← A.size
    A[k] ← x
    # Next, repeatedly swap x with the previous element, as long as they are out of order.
    # Loop Invariant: x = A[k] < A[k + 1] ⩽ ... ⩽ A[A.size].
    while k > 0 and A[k − 1] > A[k]:
        swap A[k − 1] with A[k]      # Count only this operation!
        k ← k − 1
    A.size ← A.size + 1
```

In this question, you will analyse the time complexity of algorithm SORTEDINSERT by counting only the number of *swaps* executed (ignoring all other operations).

Part (a) [2 MARKS]

What is the **best** case number of swaps executed by SORTEDINSERT? Justify your answer.

Part (b) [2 MARKS]

What is the **worst** case number of swaps executed by SORTEDINSERT? Justify your answer.

Question 8. (CONTINUED)

Part (c) [3 MARKS]

Define a sample space and a probability distribution that *could* be used to analyse the average case number of swaps executed by SORTEDINSERT.

Part (d) [5 MARKS]

Let $A = [1, 3, 5, 7]$ and choose x as follows: pick two values a, b independently and uniformly at random from the set $\{0, 2, 4\}$ and let $x = a + b$.

Compute the **average** case number of swaps executed by SORTEDINSERT(A, x), where (A, x) is the random input described above—**do NOT use your probability distribution from Part (c)**; instead, use the probability distribution defined implicitly above. Explain your reasoning, show your work, and simplify your final answer.

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

On this page, please write nothing except your name.

Last (Family) Name(s): _____

First (Given) Name(s): _____

Total Marks = 60