UNIVERSITY OF JOHANNESBURG

FACULTY OF SCIENCE

ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

MODULE	CSC03A3/CSC3A10 COMPUTER SCIENCE 3A
CAMPUS	AUCKLAND PARK CAMPUS (APK)
ASSESSMENT	SEMESTER TEST 2

DATE: 2023-05-18	**SESSION:** 14:00 - 16:00
ASSESOR(S):	PROF D.T. VAN DER HAAR PROF H. VADAPALLI
INTERNAL MODERATOR:	MR S. SITHUNGU
DURATION: 120 MINUTES	**MARKS:** 100

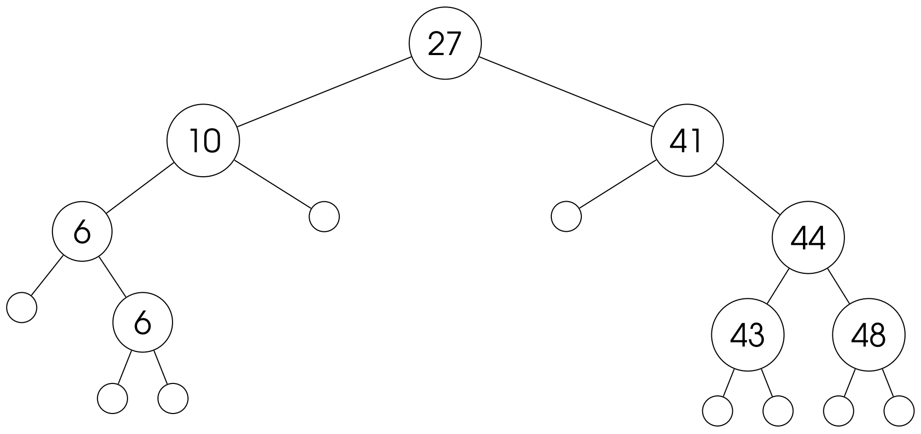
Please read the following instructions carefully:

1. You are bound by all university regulations. Please take special note of those regarding assessment, plagiarism, and ethical conduct.
2. Answer **all** the questions
3. Write *cleanly* and *legibly*.
4. You may use a non-programmable calculator to answer the questions.
5. This paper consists of 4 pages.

QUESTION 1

(a) Suppose you have a binary tree with 16 nodes. What are the minimum and maximum possible heights of the tree? You may use diagrams to aid your answer. (4)

(b) Consider the tree below, and answer the questions that follow: (16)



1. List the elements in the order of a the following traversals:
   i. PreOrder traversal (2)
   ii. PostOrder traversal (2)
   iii. InOrder traversal (2)
2. Is the above tree a Heap? (2)
3. Is the above tree an AVL Tree? (2)
4. What is the **depth** of node with key 44? (2)
5. What is the height of the whole tree? (2)
6. List the external nodes for the tree (2)

Total: 20

QUESTION 2

(a) What is the best performance your can get for the *removeMin* of a **Priority Queue ADT**, if a **array** is used as its underlying data structure. (2)

(b) Give the three (3) properties of a **total order relation**, within the context of the Priority Queue ADT. (3)

(c) Illustrate the execution of the **bottom-up construction of a heap** on the following sequence. You only need to provide a graphic representation of the heap at each stage in the construction, including any intermediate operations. (10)

3, 6, 1, 10, 7, 52, 43, 25, 16, 9, 8, 32, 2, 35, 40

(d) Discuss the **Adaptable Priority Queue ADT**, along with the operations that differ from a **Priority Queue** and their performance, when implemented with a heap. (5)

Total: 20

QUESTION 3

(a) Provide pseudo code for the **remove(k)** operation of a **List-based Map**. (5)

(b) Given a hash function $h(k) = kmod7$ for a hash table that uses **linear probing**, redraw the hash table below and **insert** the keys 84, 77, 94, 6, 30, 49, 47, 89 in this order. (8)

0	1	2	3	4	5	6	7	8	9	10	11

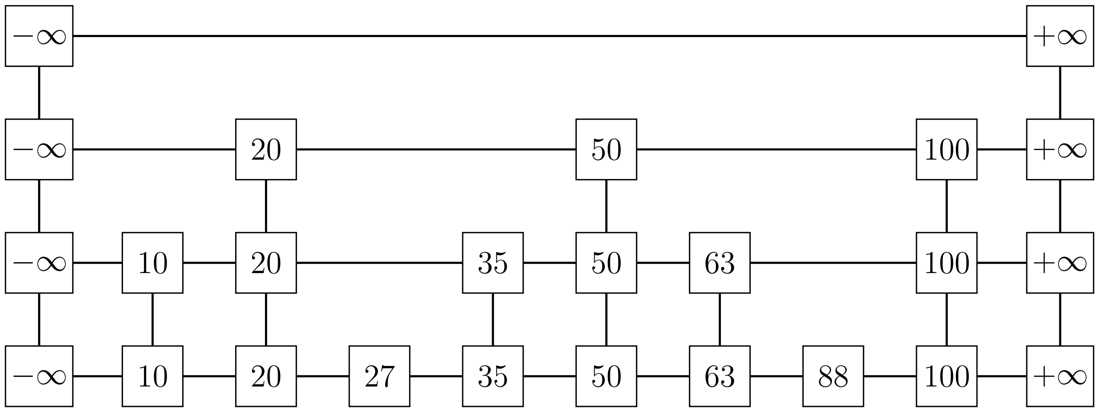
(c) What is the **load factor** for the above hash table **after** all the entries have been inserted? (2)

Total: 15

QUESTION 4

(a) Discuss how the **Search Table's** find operation achieves $O(logn)$ worst case performance. Provide an application example where a Search Table can be used effectively. (5)

(b) Analyse the skip list below and illustrate using diagrams how you would **insert** an entry with a key of 73 and 2 heads coin flips. (6)


$-\infty$
$+\infty$
$-\infty$ 20 50 100 $+\infty$
$-\infty$ 10 20 35 50 63 100 $+\infty$
$-\infty$ 10 20 27 35 50 63 88 100 $+\infty$

(c) What is the expected height of a skip list with n entries? Justify your answer. (4)

Total: 15

QUESTION 5

(a) Provide the keys steps or algorithm required for the *search(k)* function in a Binary Search Tree. (4)

(b) What is the worst case computational complexity of the following regular binary search tree operations: (3)

i. find
ii. insert
iii. remove

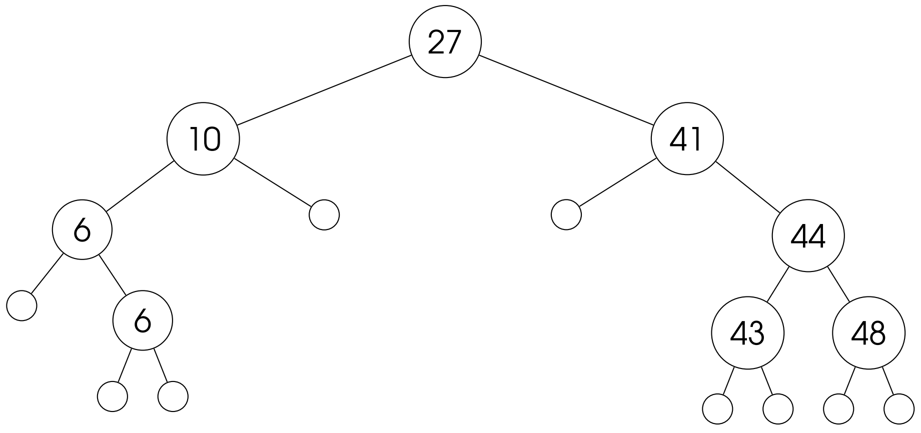
(c) Consider the regular Binary Search Tree tree below. Draw the tree state after each of the following operations. If there is a duplicate key, the inorder predecessor should be used. Removal operations should follow from the tree that resulted from the insertion operations (i.e. removals take place after all the inserts have been completed). (8)

1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order, using an inorder predecessor duplicate strategy)

   7, 33

2. Delete nodes that contain the following keys: (removed one-by-one, in the given order, using an inorder succession strategy)

   6, 44, 41



Total: 15

QUESTION 6

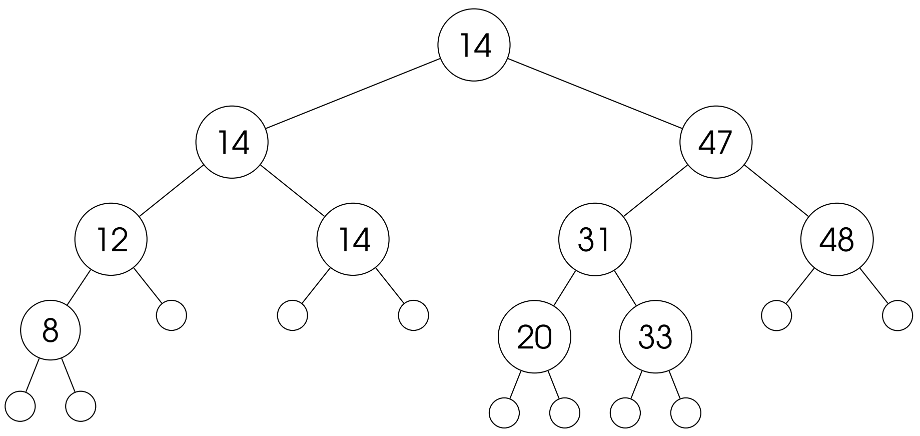
Consider the AVL tree below. Draw the AVL tree state after each of the following operations. If the tree is rebalanced draw the state before and after it being balanced. If there is a duplicate key, the inorder predecessor should be used. Removal operations should follow from the tree that resulted from the insertion operations (i.e. removals take place after all the inserts have been completed).

1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order, using an inorder predecessor duplicate strategy)

   13, 22, 27, 28, 3

2. Delete nodes that contain the following keys: (removed one-by-one, in the given order, using an inorder succession strategy)

   20, 47



Total: 15

— End of paper —