UNIVERSITY OF JOHANNESBURG

# FACULTY OF SCIENCE

## ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

**MODULE** **CSC3A10**
ADVANCED DATA STRUCTURES AND ALGORITHMS

**CAMPUS** **APK**

**EXAM** **EXAMINATION — JULY 2016**

**DATE** 2016-07 **SESSION** 12:30 – 15:30

**ASSESOR(S)** **DR DT VAN DER HAAR**

**EXTERNAL MODERATOR** **DR K NAUDE (NMMU)**

**DURATION** 3 HOURS **MARKS** 150

---

**NUMBER OF PAGES: 13 PAGES**

**INSTRUCTIONS:**

1. ALL QUESTIONS MUST BE ANSWERED
2. ANSWER THE QUESTIONS IN NUMERIC ORDER
3. CALCULATORS ARE NOT PERMITTED TO BE USED
4. WRITE CLEARLY AND LEGIBLY

**REQUIREMENTS:** NONE

## QUESTION 1

(a) Define the following within the context of object oriented design: [5]

1. Abstraction
2. Encapsulation
3. Euclid's algorithm
4. Type Erasure
5. Sentinal node

(b) Singly-Linked lists form the basis for many different types of Abstract Data Types (ADTs). Discuss how a **remove at tail** operation would be realised (you may use psuedo or Java source code to support your answer). [5]

(c) Discuss recursion, along with the four (4) different **types** of recursion used in development. [5]

(d) Write a recursive Java function that realises the following expression: [5]

$$f(n) = \begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_i = F_{i-1} + F_{i-2} \quad \textit{for } i > 1 \end{cases}$$

**[20]**

## QUESTION 2

(a) Consider the following function and using **primitive counting** express the runtime of this function in Big-Oh notation. You should also state your assumptions and which operations should be considered to be primitive. [5]

```
1  public void SS(int[] x) {
2      for (int i=0; i<x.length-1; i++) {
3          for (int j=i+1; j<x.length; j++) {
4              if (x[i] > x[j])
5              {
6                  int temp = x[i];
7                  x[i] = x[j];
8                  x[j] = temp;
9              }
10         }
11     }
12 }
```

(b) Discuss **experimental studies** and the role it plays in optimization, along with three (3) of it **limitations**. [5]

(c) Discuss how an ArrayList can accomodate an "infinite" amount of elements even though an array is of a fixed size. Name **and** describe the strategies used to achieve this. [5]

(d) Provide a discussion on the **differences** between a Node List and an Array List, along with their **performance** for *access*, *insert* and *remove* methods. [5]

**[20]**

## QUESTION 3

(a) Consider the following List Interface and write a class *Queue* that makes use of the List Interface and the **Adaptor pattern** to realise a *Queue ADT*. [10]

```
1  public interface IList<T> {
2          public Node<T> addAfter(Node<T> elem, T item);
3          public Node<T> addFirst(T item);
4          public Node<T> addLast(T item);
5          public T remove(Node<T> elem);
6          public Node<T> search(T elem);
7          public Node<T> first();
8          public boolean isEmpty();
9          public Integer size();
10 }
```

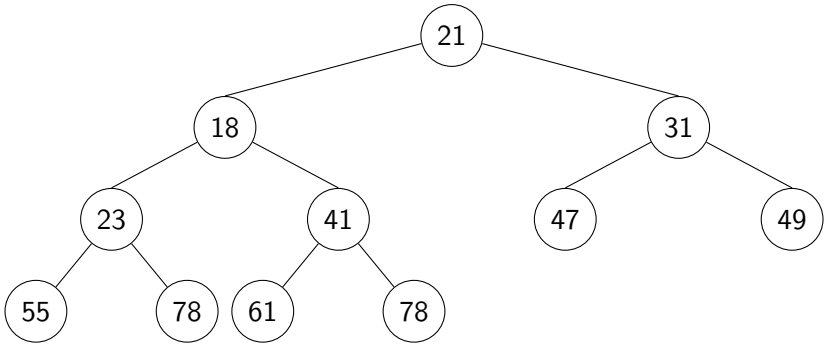
(b) Discuss the two (2) ways a Sequence-Based **Priority Queue** can be implemented, along with their **performance** for common methods. [6]

(c) Discuss the **downheap** operation found in the Heap ADT, along with its **performance**. [4]

**[20]**

## QUESTION 4

(a) Analyse the Heap key diagram below and draw the **heap state** for every step in an **insert operation** for the key value of 20. [8]




(b) Discuss the **Inorder traversal** and provide one application example where it can be used. [3]

(c) Discuss how a **Binary Tree** can be implemented using an **array** as the underlying data structure. [4]

**[15]**

## QUESTION 5

(a) Discuss how the **Map** ADT can be implemented using a **List**. Be sure to indicate how the *get* and *put* methods are implemented. [5]

(b) Discuss how **linear probing** is achieved in Hash Tables. Provide psuedo code that shows how a search [5]

(c) Name and discuss two (2) methods of handling **collisions** in a hash table, along with the role of a **load factor** in a hash table. [5]

**[15]**

## QUESTION 6

Consider the following AVL tree provided below. **Provide a graphic representation** of the AVL Tree resulting from the following operations. You must provide a **graphic representation of each step in the process**, including all intermediate operations.

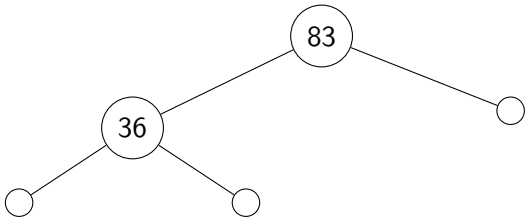
1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order)

   24, 80, 3, 34, 40, 51

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

   36, 24, 34, 40, 83, 51, 3

The AVL tree is in the current state:




The sequence of operations that result in the current state of the AVL tree can be found in Appendix A on Page 11.

**[15]**

## QUESTION 7

Consider the following 2-4 tree provided below. **Provide a graphic representation** of the 2-4 Tree resulting from the following operations. You must provide a **graphic representation of each step in the process**, including all intermediate operations. Relevant operations must make use of the **Inorder predecessor**. Removal operations should follow from the tree that resulted from the insertion operations.

1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order)

   61, 45, 24, 35, 69

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

   69, 60, 45

The 2-4 tree is in the current state (leaf nodes are not shown, however they are assumed to exist):


The sequence of operations that result in the current state of the 2-4 tree can be found in Appendix B on Page 12.

**[15]**

## QUESTION 8

Consider the following Red-Black tree provided below. **Provide a graphic representation** of the Red-Black Tree resulting from the following operations. You must provide a **graphic representation of each step in the process**, including all intermediate operations. Relevant operations must make use of the **Inorder successor**. Removal operations should follow from the tree that resulted from the insertion operations.

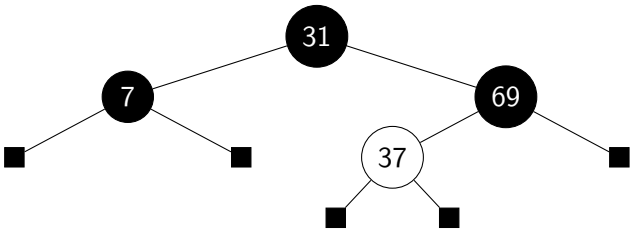
1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order)

   15, 96, 59, 40

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

   40, 69, 59, 37

The Red-Black tree is in the current state:




The sequence of operations that result in the current state of the Red-Black tree can be found in Appendix C on Page 13.

**[15]**

## QUESTION 9

(a) Discuss the various ways that a graph can be **constructed**. What is the **advantage** of each construction method? [5]

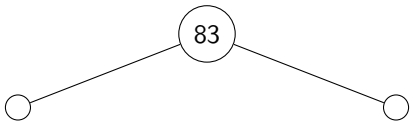
(b) Provide an algorithm that releases a **Depth First Search** of a graph. [5]

(c) Graphs and Trees are common data structures used in Computer Science problems. In many cases the choice of a Tree or a Graph is dependent on the problem being solved. Provide a discussion of the **differences** between Trees and Graphs in terms of applicable algorithms, runtime complexities, and uses. [5]

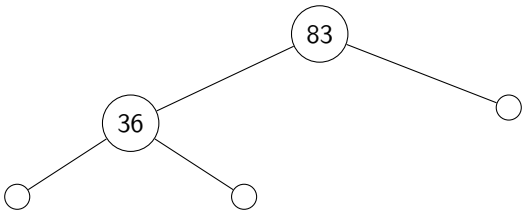
**[15]**

# A AVL Tree Operations

**Insert 83:**




**Insert 36:**



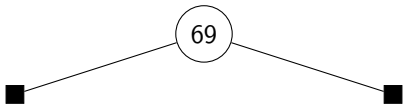
# B 2-4 Tree Operations

**Insert 40:**

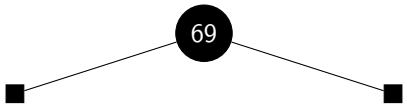

**Insert 60:**

# C Red-Black Tree Operations

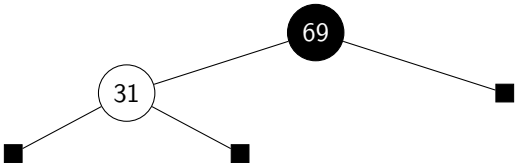
**Insert 69:**



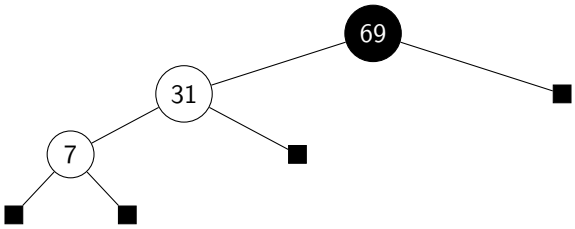

**Set Root Black:**



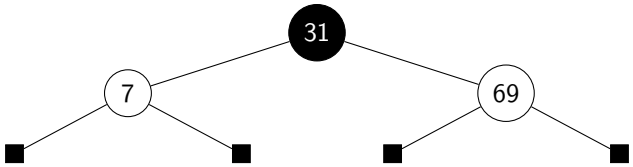

**Insert 31:**



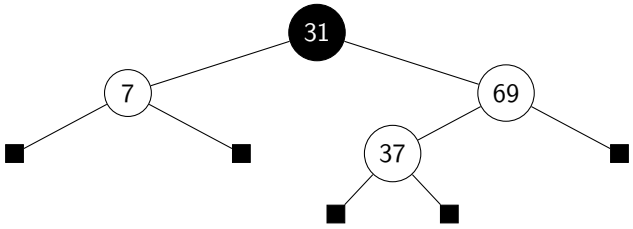

**Insert 7:**



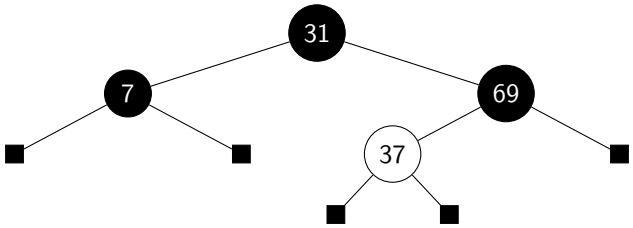

**Double Red: 1. Trinode Restructure:**




**Insert 37:**




**Double Red: 2.1 Recolouring:**




— End of exam —