UNIVERSITY OF JOHANNESBURG

FACULTY OF SCIENCE

ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

MODULE	**CSC3A10** ADVANCED DATA STRUCTURES AND ALGORITHMS
CAMPUS	**APK**
EXAM	**EXAMINATION — JUNE 2016**

DATE 2016-06-09 **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) Analyse the Java source code below and answer the questions that follow. [5]

```
public static void main(String[] args){
        T[] arr = new T[100];
        for(int i;i<arr.length;i++){
                System.out.println(i+":"+arr[i]);
        }
}
```

1. What is wrong with it?
2. Provide source code that will fix the problem.

(b) Discuss Circular Linked List data structures, along with **how** it can be used to support other ADT implementations. Provide one **disadvantage** of list based data structure in ADTs. [5]

(c) Discuss **casting** within the context of Java and the **role** it plays in object orientation. Provide Java source code to support your answer. [5]

(d) Consider the following **recursive** function and provide Java source code for a **iterative** version of the same function. You may assume all the imports have been provided and if necessary you may add additional parameters to the function. [5]

```
public static int Ack(int m, int n){
        if(m==0 || n==0)
                return m+n+1;
        return Ack(m-1, Ack(m,n-1));
}
```

[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]

```
public static void BS(int [ ] num){
      int j;
      boolean flag = true;
      int temp;

      while (flag){
              flag= false;
              for(j=0; j<num.length -1; j++){
                      if ( num[j] < num[j+1])}
                              temp = num[j];
                              num[j] = num[j+1];
                              num[j+1] = temp;
                              flag = true;
                      }
              }
       }
}
```

(b) Discuss how you would implement an **iterator** for an array-based ADT (make sure you mention which methods need to be implemented and how it will be used). [5]

(c) Discuss the **Position ADT**, along with the **benefits** and **limitations** of using it. [5]

(d) Discuss how the **principle of locality** can be used to optimise a Sequence ADT. Be sure to include the advantages **and** disadvantages of using this approach. [5]

[20]

QUESTION 3

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

```java
public interface IList<T> {
        public Node<T> addAfter(Node<T> elem, T item);
        public Node<T> addFirst(T item);
        public Node<T> addLast(T item);
        public T remove(Node<T> elem);
        public Node<T> search(T elem);
        public Node<T> first();
        public boolean isEmpty();
        public Integer size();
}
```

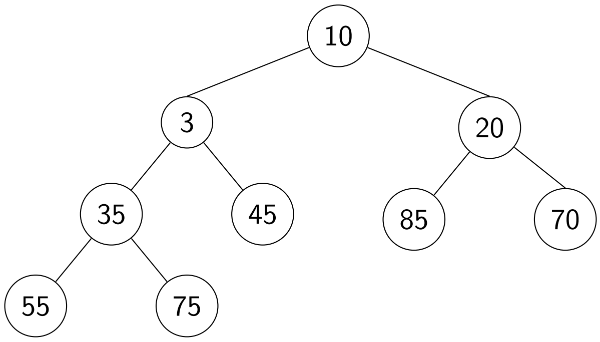
(b) Discuss the three (3) **properties** that are used to achieve **total order** relations for Entries in a Priority Queue. [6]

(c) Discuss the **upheap** 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 1. [8]



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

(c) Discuss **Decision Trees**, along with the **role** they play in certain fields, such as Artificial Intelligance. [4]

[15]

QUESTION 5

(a) Discuss how you would implement a **Dictionary** using a **Hash Table** for the underlying implementation. Be sure to include **performance** estimates for all the relevant methods in the Dictionary. [5]

(b) Discuss how an **insert** operation is performed in a Binary Search Tree. You may provide psuedo code or a diagram to support your answer. [5]

(c) Provide a discussion on **randomized algorithms**, along with the role they play in Skip Lists. As part of your discussion provide an outline of the **requirements or assumptions** of a randomized algorithm for it to work effectively. [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. Relevant operations must make use of the **Inorder successor**. Removal operations should follow from the tree that resulted from the insertion operations.

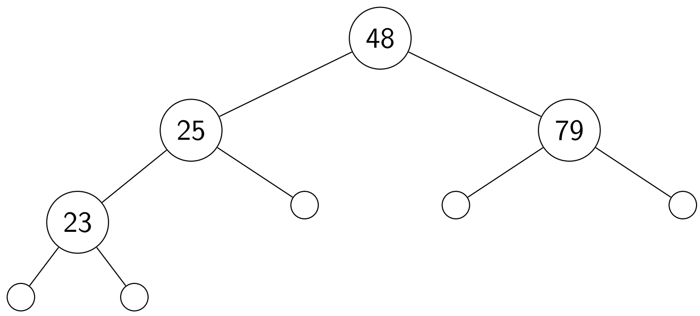
1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order)

   84, 87, 50, 4

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

   84, 50, 79, 48, 23, 25

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)

   72, 36, 5, 68

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

   72, 82, 24, 68

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

24 82 94

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.

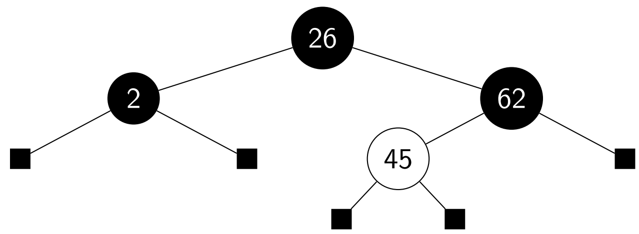
1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order)

   34, 59, 23, 93

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

   93, 45, 2, 62, 59, 23, 34

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) Provide a definition for the following components of a graph: [5]

1. End Vertices
2. Incident Edges
3. Adjacent Vertices
4. Degree of a Vertex
5. Self-loop

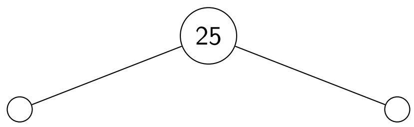
(b) Provide an algorithm that releases a **Breadth First Search** of a graph. [5]

(c) In terms of a **Graph** Data Structure outline 3 applications, with an appropriate **justification**, of a graph. [5]

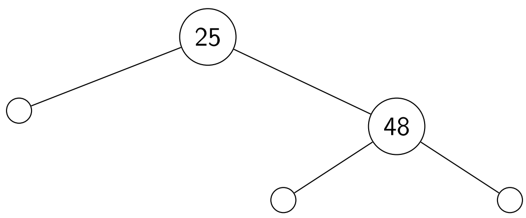
[15]

A AVL Tree Operations

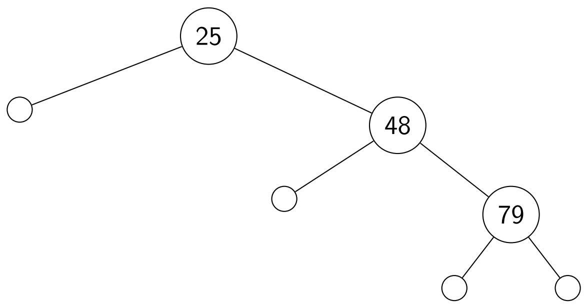
Insert 25:



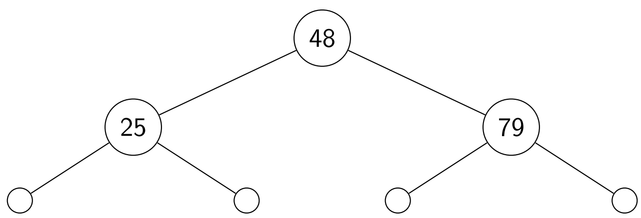
Insert 48:



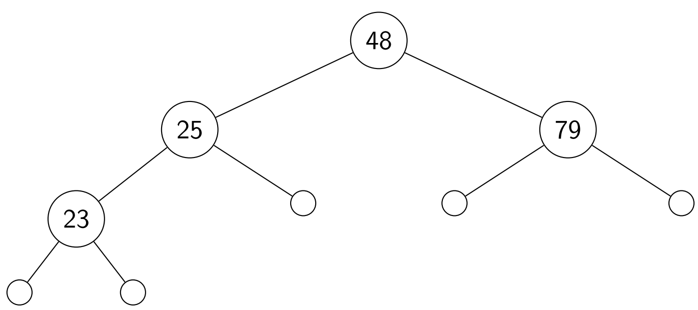
Insert 79:



Rebalancing:



Insert 23:



B 2-4 Tree Operations

Insert 82:

$\boxed{82}$

Insert 94:

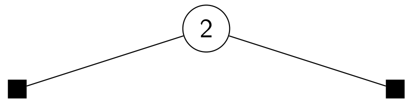
$\boxed{82\ 94}$

Insert 24:

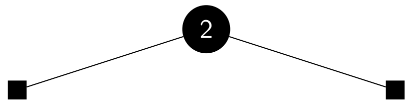
$\boxed{24\ 82\ 94}$

C Red-Black Tree Operations

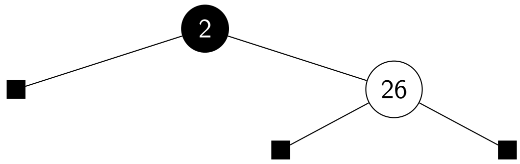
Insert 2:



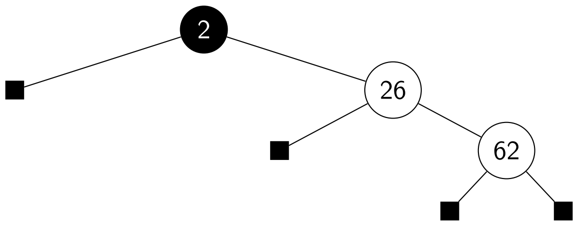
Set Root Black:



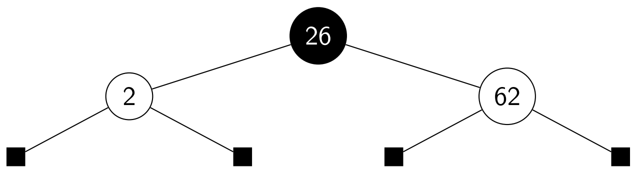
Insert 26:



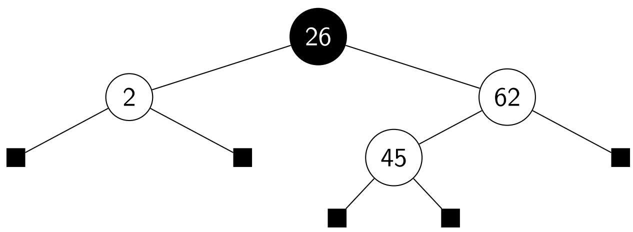
Insert 62:



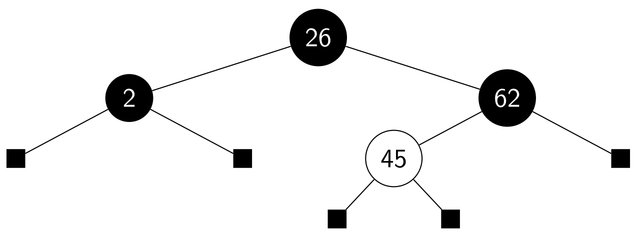
Double Red: 1. Trinode Restructure:



Insert 45:



Double Red: 2.1 Recolouring:



— End of exam —