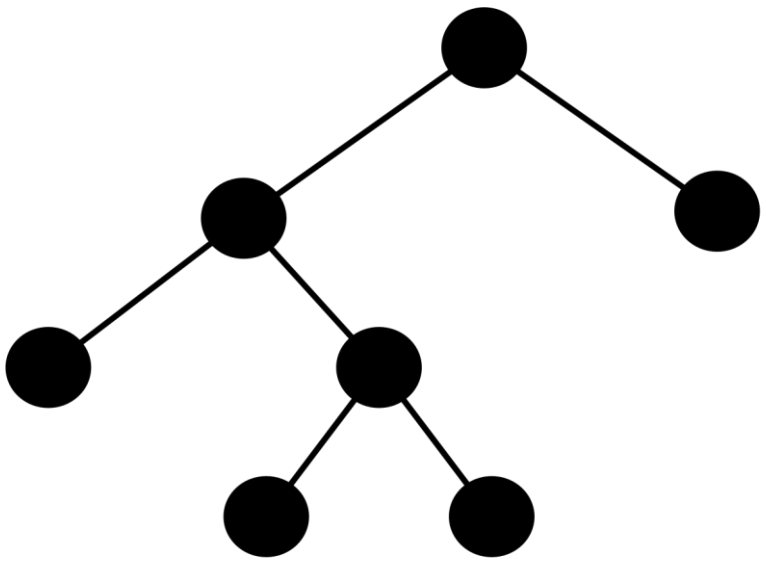
Exercise 6 – Tree Traversal (45 Marks)

This is a simple program that creates a binary tree with set values.

Study the code (shown provided below and is provided electronically) and try to understand what is happening in the program, before attempting the questions that follow.

```
1   class Node():
2       def __init__(self, value, left = None):
3           self.value = value
4           self.left = left
-
6   class Tree():
7       def __init__(self):
8           self.__rootNode = self.createBalancedTree()
-
10      def createBalancedTree(self):
11          node1 = Node(1)
12          node2 = Node(2, node1)
13          node4 = Node(4)
14          node3 = Node(3, node2, node4)
15          node6 = Node(6)
16          node8 = Node(8)
17          node7 = Node(7, node6, node8)
18          node5 = Node(5, 3, node7)
19          return node5
20
-   tree = Tree()
22  print(tree.rootNode.value)
23  print(tree.rootNode.right.value)
24  print(tree.rootNode.left.value)
25  input()
```



 © ZigZag Education, 2019

SECTION A

A 1 Give a line number from the program that contains a procedure. **[1]**

.....

A 2 Give a line number from the program that contains instantiation. **[1]**

.....

A 3 Draw the tree that is created by the program. **[2]**

A 4 The tree created by the program is a binary tree.
Explain the difference between a binary tree and a multi-branch tree. **[2]**

.....

.....

.....

A 5 The program encounters an error when trying to display the value of the root node.
Explain the cause of this error. **[1]**

.....

.....

A 6 The program encounters an error when trying to display the value of the root node's right child node.
Explain the cause of this error. **[1]**

.....

.....

 © ZigZag Education, 2019

A 7 The program encounters an error when trying to display the value of the root node's left child node. Explain the cause of this error. **[1]**

.....

.....

A 8 Explain the purpose of the code `left = None` on line 2 of the program. **[2]**

.....

.....

.....

A 9 Write the tree values as they woud be returned in a depth-first (post-order) tree traversal. **[1]**

.....

.....

A 10 Write the tree values as they would be returned in a breadth-first tree traversal. **[1]**

.....

.....

Section A: /13

 © ZigZag Education, 2019

SECTION B

B 1 Modify the program so that it does not encounter an error when it tries to display the value of the root node. **[1]**

Program updated ☐

B 2 Modify the program so that it does not encounter an error when it tries to display the value of the root node's right child node. **[1]**

Program updated ☐

B 3 Modify the program so that it does not encounter an error when it tries to display the value of the root node's left child node. **[1]**

Program updated ☐

B 4 Modify the program to add a `depthFirstSearch` function that takes a root node as input and performs a depth-first (postorder) tree traversal from that root node. The main procedure should be modified to call the `depthFirstSearch` function using the `rootNode` value from the created `tree` object. Each value should be displayed in the order in which it is checked in the tree traversal. **[3]**

Program updated ☐

B 5 Modify the program to add a `breadthFirstSearch` function that takes a root node as input and performs a breadth-first tree traversal from that root node. The main procedure should be modified to call the `breadthFirstSearch` function using the `rootNode` value from the created `tree` object. Each value should be displayed in the order in which it is checked in the tree traversal. **[3]**

Program updated ☐

B 6 Modify the `Tree` constructor and `createBalancedTree` function so that, when a tree is created, it takes a sorted list of numbers as input and creates a balanced binary search tree from that list, with `createBalancedTree` returning its root node. The main procedure should be modified to create the list `[1,2,3,4,5,6,7,8]`, construct a `Tree` object with this list, and perform both depth-first and breadth-first tree traversals on this tree. **[6]**

Program updated ☐

B 7 Modify the program to add a `binarySearch` function in the `Tree` class that takes an integer value as input, and performs a binary tree search to find the given value in a balanced binary tree, displays a message to state whether or not the given value is in the tree being searched and the number of elements that were checked, and returns the node with the given value and that node's parent node (or *None* if the value is not found). The main procedure should be modified to call the `binarySearch` function on the tree created from the list `[1,2,3,5,6,7,8]` to search for the values *1*, *2*, *4*, *5* and *9*. **[6]**

Program updated ☐

B 8 Modify the program to add a `removeNode` procedure in the `Tree` class that takes an integer value as input, searches for this value using the `binarySearch` function, and removes it from the tree. The parent of the removed node should be made to point to one of the removed node's child nodes, if it had any (if there are two, it should point to the left child). If the removed node had two child nodes, the left child node should point to the right child node. The main procedure should be modified to remove the value *2* from the tree, and then perform a binary search for the values *1*, *2* and *3*. **[6]**

Program updated ☐

B 9 Modify the program to add an `addNode` procedure in the `Tree` class that takes an integer value as input and then adds this to the tree in the correct place. Note that you will also need to create an `addValue` method in the `Node` class to traverse the tree to the correct point at which to add the `Node`. The main program should be modified to add the values *9*, *10* and *14* to the tree, and then perform a binary search for the values *9*–*15*. **[5]**

Program updated ☐

Section B: /32

 © ZigZag Education, 2019