LM Data Structures, Algorithms, and Databases (34140, 34141, 34139, 36989)

Exercise Sheet

Week 03

Q1. MCQs

1.1.

$f(n)$ is $O(g(n))$ means, intuitively, $f(n)$ is approximately proportional to $g(n)$ for large values of *n*. Which of the following statements is true?

A $5n^2 + 8n - 20$ is $O(n^2)$

B $n + \log\ n$ is $O(n)$

C $n \cdot \log\ n$ is $O(n)$

D $4n^2 - 256\,n$ is $O(n)$

1.2

$f(n)$ is $O(g(n))$ means that $f(n)$ is bounded ***above*** by some constant times $g(n)$ for large values of *n.* Which of the following statements is true?

A $8n - 20$ is $O(n^2)$

B $n + \log\ n$ is $O(n)$

C $n \cdot \log\ n$ is $O(n^2)$

D $5n^2 + 8n$ is $O(n)$

1.3 What is the worst-case time complexity for insertion in a binary search tree with n elements?

A $O(\log\ n)$

B $O(n)$

C $O(n^2)$

D $O(n\ \log\ n)$

1.4. What is the worst-case time complexity for insertion in an AVL tree with n elements?

A $O(\log\ n)$

B $O(n)$

C $O(n^2)$

D $O(n\ \log\ n)$

Q2. Determine the time complexity for the given algorithms

2.1 An algorithm which multiplies all elements in the array:

```
def product(arr):
    n = len(arr)
    x = 1
    i = 0
    while i < n:
        x *= arr[i]
        i += 1
    return x
```

2.2 An algorithm which modifies the last value in the array

```
def modify(arr):
    if len(arr) == 0:
        raise Exception("Array is empty")

    last = arr[-1]

    if last < 0:
        last = -last

    arr[-1] = last
```

2.3 Finding the largest element of the array (method 1)

```
def largest1(arr):
    n = len(arr)
    max_val = 0

    for i in range(n):
        is_largest = True

        for j in range(n):
            if arr[i] < arr[j]:
                is_largest = False

        if is_largest:
            max_val = arr[i]

    return max_val
```

2.4 Finding the largest element of the array (method 2)

```
def largest2(arr):
    n = len(arr)
    max_val = 0

    if n == 0:
        return 0
    else:
        max_val = arr[0]
        for i in range(n):
            if arr[i] > max_val:
                max_val = arr[i]

        return max_val
```

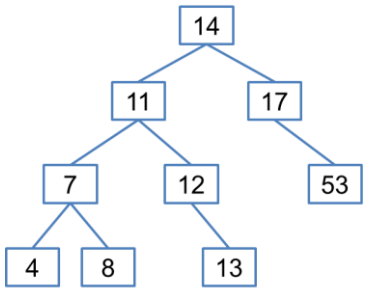
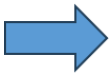
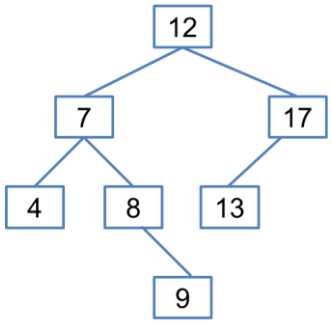
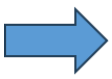
2.5 Finding the largest element of the array (method 3)

```
def largest3(arr):
    arr.sort() # assume O(n log n)

    if len(arr) == 0:
        return 0
    else:
        last = arr[-1]
        return last
```

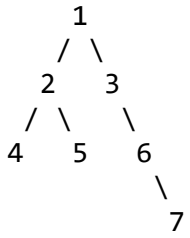
Q3. Build an AVL tree with the following values: 15, 20, 24, 10, 13.

Q4. Show the result of deleting values from the given AVL tree.

	Delete 53 	
	Delete 17 	

Q5. Create pseudocode for a function called **calculateSum** that takes the root of a binary tree as input and computes the sum of all the numbers stored in the nodes of the tree.

Q6. Create pseudocode for a function called **nodeAtLevel(tree, theLevel)**. This function should return null if the binary tree does not contain any nodes at level theLevel; otherwise, it should return the nodes present at this level.



For instance, given the following tree, when called with **nodeAtLevel(root, 3)**, the function should return [4, 5, 6].

Your task is to design the pseudocode for the **nodeAtLevel** function. What is the time complexity of your code as a function of the number of nodes in the binary tree?

Q7. Illustrate the binary tree structure with single-character data fields, given the inorder traversal output as ABCDEFGHIJ and postorder traversal output as BDCAEHGJIF.