ASSESSMENT

Python Software Stream, assessment test 2 hours

SECTION TYPE	TOTAL MARKS AVAILABLE	NOTES
Complexity Analysis	25	Multiple questions, all comprising 25 total
Algorithms 1 (Coding)	25	1 question only
Algorithms 2 (Coding)	25	1 question only
Concept Generation & Prototype (Design)	25	1 question only. Essay-based answer is needed
100 marks available total		

Please note:
This is a closed book exam - 2 hours in length.

You are allowed to use PyCharm or any other tool as you see fit (but no googling! Unless it's for looking-up documentation or syntaxes: other than that, please don't outright cheat as we will know + it's good to be ethical!).

You are given this PDF and / or Word document alongside a **Python file** (e.g. 'code environment'). This **.py** contains skeleton code for you to modify, for the algorithm questions, as well as sample test cases that you can use to test your solution!

Please leave your code in the **.py** file, and your submission for Q1 and Q4 in **either .py file** (please just document / format it nicely for easier marking) or as a **separate marked file** (e.g. word document).

Also - when submitting, make sure to rename all the files you're passing to **include your name** - this makes marking a lot easier!

Finally - best of luck! Remember, this is not a determinant of your talent. This test is simply to assess your ability, see where you can grow, as well as push you out of your comfort zone. Stay calm, deep breaths, you got this!

`Questions begin on the next page

Complexity Analysis	25 total (each question is worth 5)

Describe the average Big O complexity (both Time and Space) of the following code samples and / or problem samples (no code). Each answer is worth 5 marks (2.5 for Time, 2.5 for Space).

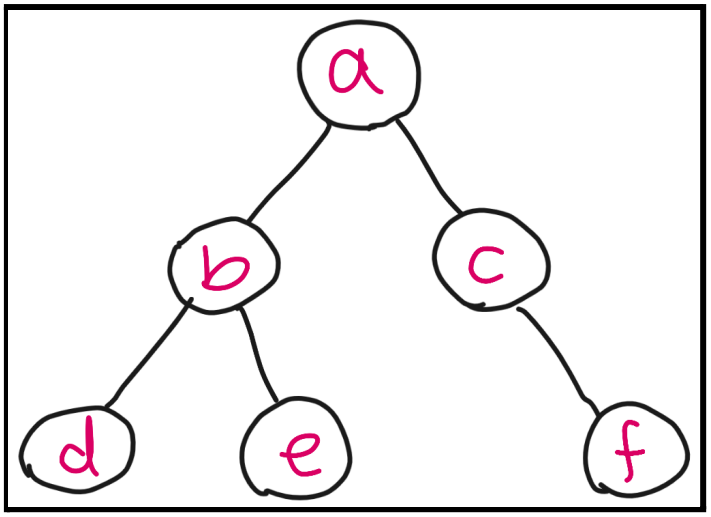
Code Sample 1 (5 marks):

```
def twoSum_HASHING(numbers, target):
    visited = {}

    for num in numbers:
        desiredNum = target - num

        if desiredNum in visited:
            return [num, desiredNum]
        else:
            visited[num] = True

    return []
```

This is a hashing-based approach for solving the Two Sum problem. Given an input array of integers and a target, it scours the array until it's able to find two numbers that add up to the target. If no such pair can be found, then it returns an empty array.

What is the Big O Time and Space complexity of this code?

Problem Sample 2 (5 marks):



I have a Graph data structure that has N number of nodes (a small sample representative of it is depicted via the image above). My hypothetical code runs a Depth First Search algorithm on it - that is, I'd like to traverse all nodes in order to store their values (e.g. add node.value to my collection). For my algorithm, I use a stack data structure to hold the order

of the nodes (e.g. visit node A, then node B, then node C). My algorithm is effectively just DFS.

What is the Big O Time and Space complexity of this code?

Code Sample 3 (5 marks):

```
def isPalindrome(string):
    left = 0
    right = len(string) - 1

    while left < right:
        if string[left] != string[right]:
            return False

        left += 1
        right -= 1

    return True
```

I regularly receive text like "abcdcba" and have devised an algorithm that can tell me whether the text is a palindrome or not. For "abcdcba" it returns False, whereas for text like "a" or "aba" it returns True (single-characters count as palindrome too!). This is a pointer-based approach to the Palindrome problem; I set a pointer to the left and another to the right, and slowly move them to the middle.

What is the Big O Time and Space complexity of this code?

Problem Sample 4 (5 marks):

No image for this problem unfortunately. Given a sorted array containing integer numbers and an integer target, I conduct a search algorithm on it in order to see if the target is present in that array or not. If it is, I return the index it was found at, else I return -1 to denote that it is not inside that array. My approach mirrors, if not is identical and effectively the same as, Binary Search. That is:

- I select the middle of the array
- I examine the middle element - if its bigger than my target, I dispose the right-half and only examine the left half (left of the target, where all the array elements are smaller)
- If it's bigger, then I do the opposite
- I repeat until there is no more to search, or my middle element matches my target
- Effectively, Binary Search
- **Hint**: *Each iteration, I dispose half of the search space / array each time. The array gets halved per iteration.*

What is the Big O Time and Space complexity of this code?

Algorithms 1 (Coding)	**25 (1 question)**

Given two non-empty arrays of integers, write a function that determines whether the second array is a subsequence of the first one.

A subsequence of an array is a set of numbers that aren't necessarily adjacent in the array but that are in the same order as they appear in the array. For instance, the numbers **[1, 3, 4]** form a subsequence of the array **[1, 2, 3, 4]** and so do the numbers **[2, 4].** Note that a single number in an array and the array itself are both valid subsequences of the array.

For example, a sample input and output would be:

Sample Input:

Array = [5, 1, 22, 25, 6, -1, 8, 10]

Sequence = [1, 6, -1, 10]

Sample Output:

true

In your answer, please discuss your solution - what is its Big O Time & Space complexity? Why have you chosen this approach? Could there be a more efficient way (and if so, how)?

If you are short on time, you can also submit pseudocode or simply describe what solution you'd write in code (just describe what you have in your mind) - this cannot attain full marks, but it is still a perfectly acceptable answer and can get partial marks.

In essence, just submit what you have even if you don't know the answer!

Algorithms 2 (Coding)	**25 (1 question)**

Write a function that takes in an array of at least three integers and, **without sorting the input array**, returns a sorted array of the three largest integers in the input array.

The function should return duplicate integers if necessary; for example, it should return **[10, 10, 12]** for an input array of **[10, 5, 9, 10, 12]**.

You may choose to submit a sorted approach, however the maximum marks this can attain is **10 marks.**

You can assume that there'll always be at least one solution - there'll always at least be a minimum of 3 numbers in the array, and you'll always be able to return a smallest to largest array output.

For example, a sample input and output would be:

Sample Input:

Array = [141, 1, 17, -7, -27, 18, 541, 8, 7, 7]

Sample Output:

[18, 141, 541]

In your answer, please discuss your solution - what is its Big O Time & Space complexity? Why have you chosen this approach? Could there be a more efficient way (and if so, how)?

If you are short on time, you can also submit pseudocode or simply describe what solution you'd write in code (just describe what you have in your mind) - this cannot attain full marks, but it is still a perfectly acceptable answer and can get partial marks.

In essence, just submit what you have even if you don't know the answer!

Concept Generation & Prototype (Design)	**25 (1 question)**

You have been tasked with creating a music application. When I (as a user) go onto this application, I should be able to select a desired song and listen to it.

You will not need to code anything, only design the application. As this is a vague and fairly open-ended question, consider the following points to include in your 'design' answer:

- Consider - who is the user? Do they have a persona? Who are you catering for?
- Design the application - what platform (web, mobile)? What may it look like (simple sketch is fine, or word-description)? What colours would you use and why? Think of this design like the way you approached the 'Design the perfect door' task for your Theory Assignment
- What design heuristics and principles will you be following? How does your design meet or fail to meet them?
- What made you choose this design?
- What are some future considerations you may need to make for your design? You can google for this question areas like "Future trends of UX / UI" (**only this area - please).**