Data Structures & Algorithms

Python Programming Assignment (35%)

**SUBMISSION REQUIREMENTS:**

1. Name the **ZIP files** containing your solutions according to the following requirements:

ZIP all Python files to be submitted into a zip file and named it as “**ADMINNO\_MiniPrj.zip**”. For example: “**123456F\_MiniPrj.zip**”.

1. At the beginning of every Python files to be submitted, include your “**Name, Student Admin no. and Tutorial Group**” as comments.
2. Submission Due Date: **7 June, Mon, 9AM (Week 8)**
3. The deliverable is to be submitted to your module tutor via Blackboard.
4. Late Submission: Marks will be deducted for late submission at the rate of 5 marks per day (out of a Maximum Marks of 35). That implies, if you submit 7 days after the due date, marks will not be given for your submission.
5. Assignment technical Review (10%) will be scheduled in week 8 during LAB session. There will be code walkthrough and technical question asked about the assignment completed.
6. **DSA Assignment (Individual – 35%)**

There are 4 components (see a to d below) to this assignment:

1. **Part 1 - Stock Inventory – Basic features (5%)**
2. **Part 1 - Stock Inventory – Search and Sort Algorithms (10%)**
3. **Part 2 - Challenge Question on Algorithm (10%)**
4. **Part 1 and 2 – Technical Review (10%)**

Part 1 is related to the development of a **Stock Inventory application**  
Part 2 is a challenge question on algorithm.

**Part 1: Stock Inventory Application**

The purpose of the Stock Inventory application is to allow the user of the application to manage the stock of a fruits store. The application will allow the user to display, add, update and remove stock items from the inventory, perform searching and sorting of stock items, and other additional features that improve the usefulness of the application for the user.

**Stock Inventory – Basic features (5%)**

1. Design a suitable data structure (e.g. by making use of Python list, dictionary, objects, etc.) to manage the stock items in the fruits store. You are required to store the following stock item information: **item index, description**, **item selling price**, **item stock level** and **at least TWO OTHER item information** of your choice. Each item will have the above attributes. The fruits store will store up to 20 different types of item.
2. Design a **menu** for the application to allow the user to perform the following:
   * + **Add** items into the stock inventory
     + **Update** selected item
     + **Remove** items from the stock inventory
     + **Display** items in the stock inventory
     + **Sort items**
     + **Search item**
     + **Exit Application**
3. Complete the (Add, Update, Remove and Display) operations for the inventory application

**Stock Inventory – Search and Sort Algorithms (10%)**

**Add the following features to the stock inventory application.**

1. Complete the Sort item option. Student must implement 3 sorting algorithms. They are **Bubble, Selection and** **Insertion Sort** algorithms to sort the items in your stock inventory based on a sort key e.g. item selling price, item stock level, etc. Student can decide on which sort key to apply.
2. Complete the Search item option. Student will apply 2 search algorithms. They are **Linear** and **Binary Search** algorithm to search for an item based on a search key. Student will decide which search key but must implement both search algorithms.
3. Student may add in bonus features such as exception handling, advanced algorithms etc

(**NOTE**: You would need to apply knowledge from the practical sessions to complete these functions)

**Part 2: Challenge Question on Algorithm (10%)**

This challenge question will test you on the design and implementation of an **efficient algorithm** to solve the following problem.

Given the following:

* 1. A sequence of positive integers, **SEQ**, sorted in strictly increasing order (i.e. all the integers are distinct), and
  2. a positive integer **Z**

Design and implement an algorithm with Python to check all pairs of integers **X** and **Y** in the sorted sequence such that **X + Y = Z**.

(**NOTE**: A brute-force algorithm that compares all possible pairs in the sorted sequence and check if they sum up to equal to **Z** will work. However, using this approach will only give you a minimal score.)

*For example, if* ***SEQ*** *and* ***Z*** *are provided to your program as follow:*

***SEQ****:*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***2*** | ***3*** | ***5*** | ***7*** | ***8*** | ***10*** | ***15*** | ***16*** | ***23*** | ***28*** |

***Z: 21***

*Then, your Python program should return:*

* ***X = 5***
* ***Y = 16***
* ***TRUE*** *(since 5 + 16 = 21)*

*However, if* ***SEQ*** *and* ***Z*** *are provided to your program as follow:*

***SEQ****:*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***2*** | ***3*** | ***5*** | ***7*** | ***8*** | ***10*** | ***15*** | ***16*** | ***23*** | ***28*** |

***Z: 3***

*Then, your Python program should return:*

* ***X = not found***
* ***Y = not found***
* ***FALSE*** *(since there does not exist two integers X and Y in SEQ, where sum of* ***X*** *and* ***Y = Z****)*

**Part 1 and 2: Technical Review (10%)**

Tutor will conduct a technical review. The purpose is to assess student’s understanding and application of DSA concepts in the project. Tutor may modify the assignment code and student may need to debug the code during the review.