Student Name: **Weight:** **15%**

Student ID: **Marks:** **/20**

Assignment: Functions, Scoping and Abstraction

Type: Group Assignment

Units: 1, 2 and 3 ONLY

- Students should **ONLY use** programming constructs covered in Units 1, 2 and 3.
- **Submissions using programming concepts that are not covered in Units 1, 2, and 3 will be penalized.**
    - **Penalty with no limit could be applied**
- **Late submissions will not be accepted.**

Scenario: Student Registration Program

An educational institution asked you to develop a student registration program. The program will have the following features:

- Storing and loading students' information into/from a file.
- Adding a new student.
- Displaying a list of registered students.
- Editing a registered student.
- Searching for a registered student.
- Deleting a registered student.
- Calculating GPA average of the registered students.

Equipment and Materials

For this assignment, you will need:

- Python IDE

Instructions

- This assignment will need to be completed outside of class time. See the course schedule and Brightspace for exact due dates.
- There are many details involved in this assignment which could be challenging to get sorted out and working correctly. Therefore, it is strongly recommended to work on the assignment as early as possible and in collaboration with your group.
- The program MUST be implemented using functions.

  - You MUST use the provided functions names.
  - You MUST implement the functions as specified.
- The program MUST use files to store/retrieve students' information.
- It is also recommended to equally divide the functions between the group members.
- You MUST use the provided TXT data file (i.e., students.txt)
- You MUST exactly reproduce the provided sample runs by running your program.
  - You MUST use the values provided in the sample runs.
  - All messages and the program menu MUST be exactly as given.
  - GPA average MUST be formatted as given.

Group Submission

- Divide the assignment into tasks and assign these tasks to members.
  - Mainly, equally divide the functions between group members.
  - Each member is responsible for implementing and testing his/her assigned functions and ensuring they are working properly.
  - Members should review the work completed by other members.
- Integrate all the functions and test the whole program to ensure that the program meets all the requirements.
- Check your solution against the detailed marking criteria at the end of this document or posted in Brightspace.
- Submit this final version of the code as a group. Only one copy is required per group, and any of the group members may submit the following to Brightspace:
  - The code of the program that you implemented (.py file).
  - A copy of your program's output from running the full test plan (.txt file).
  - A completed peer assessment document (.docx/.pdf).

Program Requirements

- The program MUST use functions to ensure reusability of the code.
- Students' information is loaded from the students text file when starting the program.
- The program should continue displaying a menu of options until the user quits the program.
  - Review the provided sample runs.
- The user chooses a menu option by entering a small or capital letter.
- When adding a new student, deleting a registered student, or editing a registered student, the file students.txt should be updated.
- The program should not allow adding a student who is already registered.
- The program should not allow deleting/editing a student who is not registered.
- The program allows editing all student's information including student id.
- Searching for, editing, or deleting should be done using student id.
- The program should have minimal code redundancy.
  - It is permitted to implement additional functions to help reduce code redundancy.
- The program should calculate the GPA average using the updated file contents.
- The program MUST implement the following functions:

Menu Option	Function	Description
	1- menu()	• It does not have parameters. • It displays the banner message and the menu options. • It asks the user to select a menu option, validates, and returns the user's selection.
	2- display_std_header()	• It does not have parameters. • It displays the student header which includes Student Name, Student ID, and GPA. • It is used when listing students and finding a student.
	3- display_student()	• It receives an index for a student and 3 lists holding student information (names, ID's and GPA's). • It displays the student header and student information associated with the index.

L/l	**4- list_students()**	• It receives 3 lists holding student information (names, ID's and GPA's).
		• If the received name list is empty, it will display "Students list has no students" error message.
		• Otherwise,
		  ○ It calls display_std_header() to display the student header.
		  ○ It iterates over the lists and displays their information.

A/a	**5- add_student()**	• It receives 3 lists holding student information (names, ID's and GPA's).
		• It asks the user to enter the student ID.
		• It checks whether the student exists or not
		  ○ If the student exists, it will display "A student with ID xxx already exists."
		  ○ Otherwise, it will ask the user to enter the additional information (name, GPA) and then append the values to the 3 respective lists.
		• After calling this function, the program should update the students text file.

E/e	**6- edit_student()**	• The program should check whether the student exists or not before calling this function. • It receives an index for a student and 3 lists holding student information (names, ID's and GPA's). • It asks the user to enter student name, ID, and GPA and updates the 3 respective lists. • After calling this function, the program should update the students text file.

D/d	**7- delete_student()**	• The program should check whether the student exists or not before calling this function. • It receives an index for a student and 3 lists holding student information (names, ID's and GPA's). • It deletes the student from the 3 respective lists. • After calling this function, the program should update the students text file.
F/f	**8- find_student()**	• It receives the student ID and the ID list. • If the student ID occurs in the ID list, it returns the index of the ID in the list. Otherwise it returns None. • For menu option 'F/f', after calling this function, the program should display the student information if the student exists.
G/g	**9- calculate_average()**	• It receives the GPA list. • It iterates over the GPA list to add up the GPA for each student and then calculates the GPA average. • It returns the calculated average.
File Handling Functions		
	10- load_students()	• It receives the text file name (i.e., students.txt) and 3 lists to hold student information (names, ID's and GPA's). • It reads student information from the file, parses and appends it into the respective lists. • Hint: the 3 lists are parallel lists. Each element in each list at the **same index** holds a different attribute of the **same student**.
	11- update_students()	• It receives the text file name (i.e., students.txt) and 3 lists to hold student information (names, ID's and GPA's). • It iterates over the lists, assembling each student's information into the required CSV format, and writing it out to the students text file.
	12- main()	• The program execution starts by calling this function. • It creates 3 lists that will hold student information (names, ID's and GPA's). • It asks the user to enter the students text file name. • It will ensure that the entered file exists before reading the file content and loading it into the lists. • It displays the program menu. • It evaluates the user selection and accordingly calls the appropriate functions.
Q/q	**End the program**.	

Test Plan

Please refer to the accompanying document, *Winter 2024 - Assignment 3 Sample Run.pdf*.

Peer Assessment

- Each member should assess the contribution/participation of all other group members.
- Each member should assign a mark out of 10 to other members.
- Ensure that the task description is clear and accurately reflects the actual participation.
- Please use the following table to complete the peer assessment. Marks/tasks are just samples.

Member/Reviewer	Member 1 Name	Member 2 Name	Member 3 Name	Completed Tasks
Member 1 Name	N/A	9	10	• Task 1 description • Task 2 description
Member 2 Name	8	N/A	9	• Task 3 description • Task 4 description
Member 3 Name	9	10	N/A	• Task 5 description • Task 6 description
Average	8.5	9.5	9.5	

A student can receive a deduction on their individual grade for the assignment if they do not meet the expected contribution/participation based on the feedback of other group members.

Marking Criteria

Rubric available in Brightspace.

