Lab #3

Spring 2025

Requirements

In this lab, you'll enhance your understanding of jagged arrays, structs, dynamic memory allocation, and file I/O by implementing a grading system that reads data from a file. The file will include the number of students for each grade on the first line, guiding the dynamic allocation process.

You are tasked with developing a grading system for a university. The system organizes students based on their letter grades (A, B, C, D, F). The first line of the file will specify the number of students for each grade, followed by individual student records. You'll implement functions to process this data, calculate averages, and filter student records based on GPA.

1.1 File Format (students.txt)

First Line: Specifies the number of students for each grade in the order A B C D F.

Example:

```
2 3 2 2 2
```

- 2 students with grade A
- 3 students with grade B
- 2 students with grade C
- 2 students with grade D
- 2 students with grade F

Subsequent Lines: Each line represents a student:

```
Rick Sanchez A 4.0

Morty Smith A 3.8

Homer Simpson B 3.0

Peter Griffin B 2.9

Bart Simpson B 3.2

Eric Cartman C 2.5

Stan Marsh C 2.7

Stewie Griffin D 1.8

Kyle Broflovski D 2.0

Meg Griffin F 1.2

Randy Marsh F 1.5
```

In this lab, you will be working with the following struct:

```c
typedef struct {
    char name[50];
    char lastname[50];
    float gpa;
} Student;
```

1.2 Jagged Array Structure

Use dynamic memory allocation to create a jagged array where, each row corresponds to a grade (A, B, C, D, F). The size of each row is defined by the first line of the input file. Each row contains first the number of students (the size), then the Student structs.

1.3 read_students_from_file

```
Student** read_students_from_file(const char* filename);
```

Info: This function will take the filename as an argument, which will read the "students.txt" file and dynamically allocate space for the jagged array. While dynamically allocating the space, you must hide the size (number of students) at the beginning as an integer. If the function cannot find/read file, the function returns NULL. If the file read is successful, it returns the Student** which is dynamically allocated by the function.

1.4 calculate_average_gpa

```
float calculate_average_gpa(Student** jaggedArray, char grade);
```

Info: This function takes Student**, which was allocated with read_students_from_file, and char of grade letter. If there are no students for the given letter grade the function return 0, otherwise the function returns the average GPA of the students with the given letter grade.

1.5 free_jagged_array

```
void free_jagged_array(Student*** jaggedArray);
```

Info: This function frees the space that was dynamically allocated for the jagged array.

1.6 list_students_above_gpa

```
int list_students_above_gpa(Student** jaggedArray, float threshold)
```

Info: This function takes Student**, which was allocated with read_students_from_file, and a float threshold of a GPA. If there is no students above the GPA, the function returns 1, if there are students above the GPA the function prints the students and returns 0.

1.7 list_students_by_grade

```
int list_students_by_grade(Student** jaggedArray, char grade);
```

Info: This function takes Student**, which was allocated with read_students_from_file, and a char of grade letter. If there is no students for the given grade letter, the function returns 1, if there are students with the given grade letter the function prints the students and returns 0.

Submission Information

Submit this assignment by using the mucsmake command.

Use the following command on Hellbender:

mucsmake <course> <assignment> <filename>

For example:

mucsmake 2050 lab3 lab3.c

Rubric: 20 points

1. Student** read_students_from_file(const char* filename);
   * 4 pts
2. float calculate_average_gpa(Student** jaggedArray, char grade);
   * 4 pts
3. int list_students_by_grade(Student** jaggedArray, char grade);
   * 4 pts
4. int list_students_above_gpa(Student** jaggedArray, float threshold);
   * 4 pts
5. void free_jagged_array(Student*** jaggedArray);
   * 4 pts

Notice:

1. All of your lab submissions **must** include documentation to receive full points.
2. All of your lab submissions must compile under GCC using the *−Wall* and *−Werror* flags to be considered for a grade.
3. You are expected to provide proper documentation in every lab submission, in the form of code comments. For an example of proper lab documentation and a clear description of our expectations, see the lab policy document.