Assignment 6: Mini Project

1 Introduction

For this assignment, we are going to write a program that allows the user to enter, display & modify student records & save them to a structure array.

The main purpose is to practise using the I/O functions in the standard C library and to practice structures with qsort.

2 The Program

This program repeatedly prompts & obtains a command (an integer) from the user & executes that command. The user can quit the program by entering the command for quitting (which is -4) or by pressing the "end-of-file" key when prompted for a command.

Note that in all user input, only the first word (if there is one) in the line the user enters is used. Extra words are simply ignored. However, the first word may need to be converted to the specific data type we need.

The following are the valid commands with brief explanations:

-4 quit
-3 sort existing records by score
-2 sort existing records by student#
-1 append a record
0 display all records
n (a positive integer) modify record number n (if it exists)

If the user enters an invalid command, it is simply ignored (with no error message) & the user prompted again for a command. Note however that the user can also quit the program using the end-of-file key when prompted for a command.

Quitting the program is obvious. Both appending & modifying a record will require inputting data for a student. The data for a student (a student record) consists of an ID & a score.

A valid ID must begin with the character 'a' followed by exactly 8 digits (e.g., a12345678); a valid score must be an integer between 0 & 100 inclusive (e.g., 65). The program needs to obtain these two pieces of information from the user & write them to the array.

Only IDs & scores are saved to the array. However, the records in the array are regarded as numbered, starting from 1. This means that the first record in the array (the first pair of ID & score) has record number 1. When a record is displayed, it is displayed with its record number. For example, when the user chooses to display all records (by entering 0 for the command), the displayed records are numbered starting from 1. The following sample output shows the output format:

```
1 a00952831 99
2 a01234567 100
3 a11112222 60
```

Note that the record number, the ID & the score are separated by a single space character. It is important that records be displayed to standard error. If there is no data to display (e.g., the user has not entered any record), the display command is simply ignored — no error message should be printed.

The numbering of records also applies when the user wants to modify data. When the user enters a positive integer for a command, that integer is regarded as the record number of the record to modify. If that record does not exist, the command is simply ignored (with no error message) & the user re-prompted for another command. If the record exists, it is displayed, again to standard error & with a record number (which should be the same as the number the user entered), before the user is prompted to enter new data for the student.

3 Entering Data for a Student

This is necessary when the user chooses to append or modify a record.
To get the data for a student, the program repeatedly prompts the user for an ID until either

1. a valid ID is obtained, or
2. the user presses the end-of-file key, or
3. the user enters -1 (i.e., minus one)

Cases 2 & 3 signify that the user wants to abort the operation. When either happens, the program goes back to prompting the user for a command. After obtaining a valid ID, the program then repeatedly prompts the user for a score until either

1. a valid score is obtained, or
2. the user presses the end-of-file key, or
3. the user enters -1 (i.e., minus one)

Again, cases 2 & 3 signify that the user wants to abort the operation. When either happens, the program goes back to prompting the user for a command.
If both a valid ID & a valid score are obtained, they are written to the array (in the appropriate location).

When modifying a record, the new information overwrites the existing record that we want modified; if appending a record, the new record is written at the end. After writing the record, the program goes back to prompting the user for a command.

Here is the record struct definition:

```
typedef struct record record;
struct record {
        char id[10];
        int score;
};
```

4 Additional Requirements

• Make sure submit your code (.c file) before the due into D2L dropbox
• Make sure your code can be compiled. If not, you will get a zero on this assignment.
• Use separate functions & do not use “global” variables.
• All prompts should be printed to standard output. Records should always be displayed to standard error. This is to separate the two types of output to facilitate testing. Note that error messages are not needed when the user enters invalid input. Hence error messages, which are also printed to standard error, should be rare.

[Hint: how to print to standard error: fprintf(stderr, "this is an error");]

• Format your code properly. You’ll lose marks if your code is not indented properly.