Assignment 3 – cScheduler

Concepts required: Files, functions, 1D and 2D arrays and strings

1.0 Contextual Situation: Course Allocation

Congratulations! You have been hired as a developer for the Computer Science to help the department with course assignment for the current year. As a developer for this software (cScheduler), you have been selected to complete the following tasks below to be used within the system.

There are currently 10 courses that are offered this year. The system tracks various info such as the number of courses taught by a professor this year and other info that is detailed in the next section.

1.1 Prerequisite – First things First

As the sole software developer of this product, you will create a program that will allow x number of courses that are taught by y number of professors. All data will be read in from external files and stored in arrays. The data files provided to you will contain data for ten (10) courses that are taught by six (6) professors.

The software will take the input data parsed from files and determine some statistics on the courses taught. For example, this program will predict the professor who will teach a particular course in the next term based on state-of-the-art (not really ☺) algorithms.

This assignment stresses the importance of developing software that strictly conforms to specification—when you're in the industry, you'll face all sorts of strict clients.

There are two source (2) files that you will need to submit:

(1) You will write a .c file, lastnameFirstnameA3.c containing the function implementations detailed below

(2) You will write a .c file, lastnameFirstnameA3Main.c with the main function

---

Include the given header file called givenA3.h in your C files – this file contains the proper function prototypes, constants, and structures required for this assignment, also detailed below.

You are also free to create helper functions, though if they are used by the assignment functions below, include them in the appropriate files so they are accessible to us during grading.

These constants, and function prototypes are defined in the given givenA3.h header file.

- `#define NUMBER_PROFS 6`
  This represents the number of professors.

- `#define NUMBER_COURSES 10`
  This represents the number of courses.

- `#define MAX_STR 50`
  This will represent the maximum length of a string.

****All Function prototypes described below are also given in givenA3.h ****

2.0 Your Task

Part A: Write function definitions for the following tasks.

```
(Task 1)

int readCourse (
      char filename [MAX_STR],
      char courseName [NUMBER_COURSES][MAX_STR],
      int courseID [NUMBER_COURSES]
);
```

This function must read courseName and courseID from a text file called courses.txt and use them to populate the array courseName with course names and courseID with corresponding course ids. A sample courses.txt file is provided. Note that the first 10 lines of this file store 10 course names, and the next 10 lines store 10 course IDs, assuming that NUMBER_COURSES is set to 10. Therefore, this function will read in exactly 2 times NUMBER_COURSES lines. Also note that course names may have multiple words in them (e.g., Introduction to Programming). All course IDs are integers.

The function will either return 1 (to indicate successful operation) or -1 to represent abnormal exit, which can be:

- Passing a NULL file pointer to the function

Make sure you test this thoroughly—this and the next function (readProfs) represent the backbone of this application. Some implementations of line-by-line file reading may inadvertently introduce an extra newline ('\n') in the string returned to the developer, which will mess up our testing. This is an error that can become "consistent" throughout the program (for example, if you mistakenly read the strings as {" Programming\n", "Introductory Programming\n", "Object Oriented Programming\n", "Data Structures\n", "Software Systems

Development and Integration\n", "Intro to Databases\n", "Analysis and design of Algorithms\n", "Advanced OO\n", "Data Mining\n", "Cyber SecurityProgramming\n"}, the rest of the program will likely run without error, only to fail the string comparison tests in grading. This is because the remaining functions will read and write the botched course names and assume they are correct.

(Task 2)

```
int readProfs (char filename [MAX_STR],
               int courseID [NUMBER_COURSES],
               char profName [NUMBER_PROFS][MAX_STR],
               int coursesTaught [NUMBER_PROFS][NUMBER_COURSES]
 );
```

The prof names and course assignments **MUST** be read from a text data file whose name is stored in `filename` (parameter# 1 of this function). A sample data.txt file is provided. The first 6 lines of the file contain the prof names (given that NUMBER_PROFS is set to 6) – these are read from the data file and saved in `profName (`parameter# 3 of this function)`.` The next 6 lines will consist of digits 1 or 0, 1 indicating that a course is taught by a prof and 0 indicating that it is not. These values 1 or 0 must be used to populate array `coursesTaught` (parameter# 4 of this function), Assume that the order of the course IDs used in this function is the same as that given in array `courseID (`parameter# 2 of this function). For example, if the **first 7 lines** of file data.txt are given as

Ritu
Manav
Ben
Ricardo
Cathrine
Sooraj
1100010000
……

the first 6 are prof names, and the 7th line implies that prof Ritu teaches courses 1300, 1500 and 3530. Accordingly, `profName[0]` is populated as `"Ritu"` and `coursesTaught` is populated as `{1300, 1500, 0, 0, 0, 3530, 0, 0, 0, 0}.` Note that in array `coursesTaught,` 0 indicates that the course is not taught by that prof; whereas a non-zero value indicates the courseID of the course taught by that prof.

For convenience, below is given a table that corresponds to the prof names and courses taught that are given in the data file called data.txt.

	1300	1500	2430	2520	2750	3530	3490	4450	4600	4750
Ritu	1	1	0	0	0	1	0	0	0	0
Manav	0	0	1	1	0	0	0	0	0	0
Ben	1	0	0	0	0	1	0	1	0	0
Ricardo	0	1	0	0	0	1	1	0	0	0
Cathrine	0	0	0	0	0	0	1	1	1	1
Soooraj	1	1	0	0	0	0	0	0	0	0

The function will either return 1 (to indicate successful operation) or -1 to represent abnormal exit, which can be:

- Passing a NULL file pointer to the function

```
(Task 3)

int nCourses (
      int n,
      char profName [NUMBER_PROFS][MAX_STR],
       int coursesTaught [NUMBER_PROFS][NUMBER_COURSES]
);
```

This function takes as input a value of n and returns the total number of professors who teach n or more courses. It also prints the professor names for professors who teach n or more courses.

For example, if course assignment is as follows and n = 3, then the function prints the following names: “Ritu”, “Ben”, “Ricardo” and “Cathrine”, and returns a value 4.

	1300	1500	2430	2520	2750	3530	3490	4450	4600	4750
Ritu	1	1	0	0	0	1	0	0	0	0
Manav	0	0	1	1	0	0	0	0	0	0
Ben	1	0	0	0	0	1	0	1	0	0
Ricardo	0	1	0	0	0	1	1	0	0	0
Cathrine	0	0	0	0	0	0	1	1	1	1
Soooraj	1	1	0	0	0	0	0	0	0	0

```
(Task 4)

int getCourseName (
       int courseNum,
      char cNameFound [MAX_STR],
      char courseName [NUMBER_COURSES][MAX_STR],
      int  courseID [NUMBER_COURSES]
);
```

This function, takes a course number as input, searches for its course name and stores it in a string parameter (e.g. cNameFound). It returns 1 if the course is found, 0 otherwise. For example, if the course number is 1300, it will return 1 and store “Programming” in cNameFound.

```
(Task 5)

int getCourseNum (
      char cName [MAX_STR],
      int * cNumFound,
      char courseName [NUMBER_COURSES][MAX_STR],
      int courseID [NUMBER_COURSES]
);
```

This function, takes a course name as input, searches for its course number and stores it in an output int parameter (e.g. cNumFound). It returns 1 if the course is found, 0 otherwise. For example, if the course number is 1300, it will return 1 and store 1300 in *cNumFound. Remember to use fgets since some course names have multiple words. Also a reminder that fgets includes the \n character!

```
(Task 6)

int profsTeachingCourse (
      int courseNum,
       char profName [NUMBER_PROFS][MAX_STR],
      int coursesTaught [NUMBER_PROFS][NUMBER_COURSES]
);
```

This function takes a course number as input, and prints names of all profs teaching it. It returns the total number of profs teaching the course `courseNum` if the course is found, 0 otherwise. It also returns 0 if the course is not taught by any prof. For example, if the course number is 1300, it prints “Ritu”, “Ben” and “Sooraj”. The function returns 3 in this case.

```
(Task 7)

float avgNumCourses (int coursesTaught [NUMBER_PROFS][NUMBER_COURSES]
 );
```

This function returns the average number of courses taught by a professor in the given course allocation scheme. Assume that the total number of professors is `NUMBER_PROFS`.

	1300	1500	2430	2520	2750	3530	3490	4450	4600	4750
Ritu	1	1	0	0	0	1	0	0	0	0
Manav	0	0	1	1	0	0	0	0	0	0
Ben	1	0	0	0	0	1	0	1	0	0
Ricardo	0	1	0	0	0	1	1	0	0	0
Cathrine	0	0	0	0	0	0	1	1	1	1
Soooraj	1	1	0	0	0	0	0	0	0	0

For the above data, the average number of courses taught by a Prof is 17 / 6 = 2.83.

(Task 8)

```
void hhistogram (char profName [NUMBER_PROFS][MAX_STR],
                 int coursesTaught [NUMBER_PROFS][NUMBER_COURSES]
);
```

This function displays a horizontal histogram showing the number of courses taught by each professor, where each course is represents by one star. The total number of courses taught must be printed at the end of the line. For example, for the following course allocation for 6 professors, this function will display a histogram as shown below (Hint: Use \t to align the stars).

```
Ritu:       *** (3)
Manav:      ** (2)
Ben:        *** (3)
Ricardo:    *** (3)
Cathrine:   **** (4)
Sooraj:      ** (2)
```

	1300	1500	2430	2520	2750	3530	3490	4450	4600	4750
Ritu	1	1	0	0	0	1	0	0	0	0
Manav	0	0	1	1	0	0	0	0	0	0
Ben	1	0	0	0	0	1	0	1	0	0
Ricardo	0	1	0	0	0	1	1	0	0	0
Cathrine	0	0	0	0	0	0	1	1	1	1
Soooraj	1	1	0	0	0	0	0	0	0	0

Part B: You must write a main program that displays a menu of choices as shown below. Call this program `lastnameFirstnameA3Main.c`. You may use the main function given in file `givenMain.c` on Courselink as the starting code for writing your main function (Don't forget to change the filename before submitting).

Note that

- The main program must first run tasks 1 and 2, so that the required arrays are populated. It must then have a **menu-driven** code that displays a menu for task 3, 4, 5, 6, 7, 8 described in section 2.0 Part A.

- **Menu choices** are shown next:

    1. Task 3 – find the total number of professors who teach n or more courses
    2. Task 4 – find course name, given its course number
    3. Task 5 – find course number, given its course name
    4. Task 6 – find all professors teaching a given course
    5. Task 7 – find the average number of courses taught by a professor
    6. Task 8 – display a horizontal histogram showing the number of courses taught by each professor

3.0 Program Submission and Expectation

The following section outlines what is expected when submitting the assignment and other information.

3.1 Program Submission

You must submit 2 files `lastnameFirstnameA3.c` and `lastnameFirstnameA3Main.c`
Incorrect file names will result in penalty.

3.2 Program Expectations

Your program is expected to follow the outlined information exactly. Failure to do so will result in deductions to your assignment grade.

- The program files you submit MUST compile with NO warnings and errors using the flags - std=c99 and -Wall.
- If you decide to define and use helper functions (other than what is listed in the .h file), then you must also add their prototypes in lastnameFirstnameA3.c, along with their definitions.
- **Programs that fail to compile will be given a mark of 0.**
- Programs that produce warnings upon compilation will receive a deduction of 1 mark for each type/category of warning.
- Penalties will occur for missing style, comments, header comments etc.
- Both program files must contain the header comment shown in section Program Header Comment Format with the **underlined & bolded** sections properly filled in.

3.3 List of "Do Not"s

- Do not change any given prototype
- Do not use Global Variables
    - Using global variables will result in 0 in the assignment

- Do not use GOTO statements
    - Using Goto statements will result in 0 in the assignment

3.4 Header Comment

Use the template given below for header comment – **you must include them in both c files you submit.**
/!\ Note: The file name, student name and email ID must be changed per file and per student.

```
/************************chaturvediRituA3.c************** Student Name: Ritu Chaturvedi Email Id: ritu

   Due Date: November … Course Name: CIS 1300

   I have exclusive control over this submission via my password.
   By including this statement in this header comment, I certify that:

   1) I have read and understood the University policy on academic integrity. 2) I
   have completed the Computing with Integrity Tutorial on Moodle; and 3) I have
   achieved at least 80% in the Computing with Integrity Self Test.

   I assert that this work is my own. I have appropriately acknowledged any and all
   material that I have used, whether directly quoted or paraphrased. Furthermore, I
   certify that this assignment was prepared by me specifically for this course.

*********************************************************/
```

The program file must contain instructions for the TA on how to compile and run your program in a header comment.

/!\ Note: The file name must be changed per student.

```
/*********************************************************
Compiling the program
The program should be compiled using the following flags: -std=c99 -Wall

compiling:
gcc -std=c99 -Wall chaturvediRituA3.c chaturvediRituA3Main.c

Running: ./a.out


OR

gcc -std=c99 -Wall chaturvediRituA3.c chaturvediRituA3Main.c -o assn3

Running the Program: ./assn3
*********************************************************/
```