Assignment #2: Arrays (40 points)

This is a continuation of program #1. This adds reading of a course definition file into an array. When the student’s course registration requests are processed, we will modify that courses information (decrease the number of available seats). **You will need to modify your cs1713p1.c program**. Copy it to cs1713p2.c

Command:

register -c courseFileName -s studentRegistrationFileName

Input:

Course Stream input file which contains many records defining courses:

szCourseId szRoom szDays szTimes iAvailSeats dFee

12s 15s 8s 15s 4d 10lf

Student (same as in Program #1) Stream input file which contains many student records, each containing possibly many course registration requests. There are three different kinds of lines of data for each reservation:

* Student Identification Information:
  + One line per request (separated by spaces)
  + szStudentId cGender szBirthDate szFullName

7s 4s 11s 31s (may contain spaces)

* + Although szFullName is a maximum of 30 characters, it may contain spaces; therefore, you cannot simply use %30s. You will have to use a bracket format code using ^\n
* Student Record Information
  + One line per reservation request (separated by commas)
  + szMajor szEmail dGpa cInternationalStudent

4s 31s 4d 1c

* Registration Request:
  + 0 to many registration requests per student (terminated by END in the course ID)
  + szCourseId

12s

Files provided:

cs1713p2.h – contains new definitions for Course struct

p2Courses.txt

p2StudentReg.txt

Process (green highlights show what you did previously) :

1. Read the Course file's data into an array of Courses (see the Course typedef in cs1713p2.h).
2. Print the current course definition information before processing any student registration requests. As shown at the end of the file.
3. (Most of this step was done in Program#1 except the actual processing of a registration request.) Read a file of Customer Reservation Request Data until EOF. For each reservation request:

* Read a data line containing the students’s id, gender, birthdate, and full name.
* Read a data line containing their major, email, gpa, and international status.
* Read possibly many (could be none) course requests (until a request is read with END in course Id).
* Process the reservation request (see below)

1. Print the course array after all registration requests have been processed. You should notice a decrease in the available seats for courses that satisfied a student registration request.

**Processing for Each Registration Request**:

* Print the student’s information.
* For each registration request:
  + Print the course ID.
  + If the course ID doesn't exist in the array of courses, show a message indicating that it was not found and continue with the next registration request. (Do not terminate.)
  + If there aren't enough seats available to satisfy the request, show a message indicating "insufficient seats" and do not decrease the number of available seats for that course.
  + Otherwise print the fee
  + Add that cost to the total cost for the registration request.
* After handling the list of course requests for a student, print the total fees for the registration request (not including any requests that could not be satisfied) as:

**total fees: $105.00**

Error Processing:

* There are some expected warnings (e.g., insufficient seats, course ID not found) which should be printed to stdout and not cause execution to terminate.
* Errors such as bad filename and incorrectly formatted input file should cause the program to be exited (use exitError) as was done in Program #1.

Modifications of Program#1:

1. Above the main function:

* change from including "cs1713p1.h" to "cs1713p2.h"
* add this declaration:

FILE \*pFileCourses; // stream input for Course Definitions

* change the prototype declaration for processCommandSwitches to

void processCommandSwitches(int argc, char \*argv[]

, char \*\*ppszStudentFileName

, char \*\*ppszCourseFileName);

1. Modify the main function as necessary.

* Declare Course courseM[MAX\_COURSES];
* Declare a variable for the count of the number of entries in the courseM array.
* Declare a variable for the course file name:

char \*pszCourseFileName = NULL;

* Just like the code for opening the Student file, add code to check the Course file name and open the Course file.
* Invoke getCourses, passing the courseM array. It will populate the courseM array and functionally return the count of the number of entries in the courseM array.
* Print the original course information before processing the registration requests.
* Invoke processRegistrations, passing the courseM array and the count of the number of entries in that array
* Print the course information after processing the registration requests
* Anything else?

1. processCommandSwitches needs to be modified to return multiple filenames (one for the student registration request file and one for the course file). (See the prototype declaration above.)
2. Add a new function for retrieving the course data. That function must also return the count of courses as its functional value:

int getCourses(Course courseM[])

1. Modify processRegistrations to receive the array of courses and course count as parameters. processRegistrations also needs to **actually process the registration requests**.

void processRegistrations(Course courseM[], int iCourseCount)

1. Add a new function printCourses which is passed an array of courses and the course count as parameters. It should print the courses in a table format with column headings and the course information (Course Id, Room, Days, Times, Available Seats, Fee).

void printCourses(Course courseM[], int iCourseCount)

1. exitUsage should be updated to show what switches are now needed.
2. Add other functions that help improve modularity.

Grading:

* Your program must be written according to my programming standards.
* Your program should use the provided include file "cs1713p2.h".
* ~~We will show sample output in class.~~
* You must turn in your .c file and your program output which is based on the input file that I provide.
* **Modularity matters. There may be additional functions (which I did not explicitly mention) that should be added to improve reuse and improve clarity.**
* Code which is mostly not working will receive less than 50% credit.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Courses \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Course ID Room Number Days Times Seats Fees**

**HIS1043.002 MH2.102 MWF 1:00-1:50pm 120 35.00**

**GEO1013.005 MB1.101 TR 12:30-1:45pm 5 35.00**

**MAT1214.003 MS1.02.03 TR 2:00-3:15pm 1 35.00**

**CS1713.002 NPB1.202 MWF 1:00-1:50pm 0 50.00**

**MAT3013.001 MS1.02.07 TR 2:00-3:15pm 1 35.00**

**ENG1023.001 MH2.202 MWF 10:00-10:50am 15 35.00**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Student Registration Request \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**pte357 Pete Moss (M 1992/01/01)**

**CS pte357@my.uni.edu INTL:N GPA:3.21**

**Course**

**HIS1043.002**

**GEO1013.005**

**MAT1214.003**

**CS1713.002 course is full not adding the fees**

**total fees: $105.00**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Student Registration Request \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**agt352 Pop Corn (M 1999/02/02)**

**MAT agt352@my.uni.edu INTL:N GPA:2.8**

**Course**

**MAT3013.001**

**MAT3233.002 course Id does not exist**

**HIS1053.004 course id does not exist**

**total fees: $35.00**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Courses \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Course ID Room Number Days Times Seats Fees**

**HIS1043.002 MH2.102 MWF 1:00-1:50pm 119 35.00**

**GEO1013.005 MB1.101 TR 12:30-1:45pm 4 35.00**

**MAT1214.003 MS1.02.03 TR 2:00-3:15pm 0 35.00**

**CS1713.002 NPB1.202 MWF 1:00-1:50pm 0 50.00**

**MAT3013.001 MS1.02.07 TR 2:00-3:15pm 0 35.00**

**ENG1023.001 MH2.202 MWF 10:00-10:50am 15 35.00**