Final Project

Project Description:

It is required to develop a program that operates on courses, exams and students data. The program uses two defined string formats one to represent courses exams data and the other to represent students data. An example of the two strings format is as follows:

CoursesExams=[101,28/4/2016,A;201,3/5/2016,B]
Students=[191001,201,101;191002,201]

Courses Exams Data

The CoursesExams string contains a list of courses' exams data separated by ';'.
Fields of a course exam are:

1. Course Code Number (ex: 101)
2. Exam Date (ex: 28/4/2016)
3. Exam Hall (ex: A) //assume all hall names consist of one alphabetic letter.

Fields of the same course exam are separated by comma as shown in the above example.

CoursesExams string is guaranteed to be no longer than 500 characters.

Students Data

The Students line contains a list of students data separated by ';'.
Fields of a students are:

1. Student ID (ex: 191001)
2. Courses' codes of student's attended courses (ex: (201,101)).

Fields of the same student are separated by comma as shown in the above example.

Student ID is guaranteed to be no longer than 6 characters.

List of students' attended courses is guaranteed to be no longer than 20 characters.

Students string is guaranteed to be no longer than 1000 characters.

Operations:

When the program starts, the user enters one CoursesExams Line and one Students line in the defined above format then followed by one or more operations from the below table (each operation in a line). The program ends when it reads **Quit** operation.

	Operation	Action Required
1	**Number_Students**	Print the number of students.
2	**Number_Halls**	Print the number of Halls
3	**Student_ID_Min**	Print the minimum student ID value
4	**Students_Dropped_ID**	Print the IDs between minimum and maximum student ID that are not assigned to any student
5	**Exams_Period_InDays**	Print the number of exams days from the start date to the end date.**(including the start and end days)**
6	**Course_Students** CourseCode	List all student IDs of the students attending the given CourseCode (CourseCode will be a valid Course Code) (every output student ID should be printed in a separate output line)
7	**List_Course_Students_More** n	List course codes of courses

		having more than n students (n will be a valid non negative integer) (every output course code should be printed in a separate output line)
8	**List_Student_Courses_Less** n	List all student IDs of the students attending less than n courses (n will be a valid non negative integer) (every output student ID should be printed in a separate output line)
9	**List_Hall_Students** HallName,Date	List all student IDs of the students attending an exam at hallName at exam Date. (HallName will be a valid hall name & Date will be a valid date)
10	**List_Hall_Students_InAnyday** HallName	List all student IDs of the students attending an exam at HallName at any date. (HallName will be a valid Hall Name) (every output student ID should be printed in a separate output line) **Repeated ID should only be printed one time**
11	**Quit**	End program

Input / Output Samples:

Assuming the user entered:

CoursesExams=[101,28/4/2016,A;201,3/5/2016,A;110,5/5/2016;103,5/5/2016,A;120,6/5/2016,D;132,7/5/2016,B]

Students=[191001,(201,101,110);191002,(201,110,103);191003,(110,120,132);191006,(103,132)]

The following are output samples of the program interaction with different commands (Input command is in blue and expected output is in black for clarification):

Number_Students
4

Number_Halls
4

Student_ID_Min
191001

Students_Dropped_IDs
191004
191005

Exams_Period_InDays
10

Course_Students 201
191001
191002

List_Course_Students_More 1
201
103
110
132

List_Student_Courses_Less 3
191006

List_Hall_Students A, 5/5/2016
191002

191006

List_Hall_Students_InAnyDay A
191001
191002
191006

Quit
Thanks!

Note: Quit should end reading any additional operations and end your program.

Hints:

While parsing data, you will need to convert from string to a number. Use atoi() function defined in <stdlib.h>.

General Constraints:

1. The user can input up to 20 courses and up to 50 students.
2. One student can attend up to 5 courses.
3. **All your code should be in one file.**
4. **Do not prompt the user to enter anything , just read the input directly and print the output of operations directly.**
5. At any operation if the output is empty the operation should print “none”
6. Output should not include any extra white spaces or any extra texts more than the results.
7. Do not clear the screen after every operation.
8. Only submit the source code file (.c file) on moodle (Do NOT submit the whole project or a zipped folder of any kind).

Bonus:

Reading the input string from a file instead of directly reading it from the console.

Grading Rubrics:

Specifications: The program works and meets all of the requirements (55%).

- Operations from 1 to 5 (35%)
- Operations from 6 to 11 (20%)

Readability: The code is well organized and easy to follow (15%).

Documentation: The code is well documented and clearly explained(20%).

Delivery: The program was delivered on time(10%).

Submission:

Each project submission must include:

1) Code in one (.c) file.
2) A report with the following:
    a. Team member names and IDs
    b. Application description
    c. Flowchart of execution sequence
    d. Sample input and output screens.

Due Date:

6/6/2020

Teams:

Work in Groups of 5 students (No exceptions accepted).

Plagiarism:

Plagiarism is a serious academic offence and students who share code with others will fail the course. A plagiarism detection tool will be used to check all projects submitted and report plagiarism cases.