Structured Programming Final Project

Subject:

This project will create a small-scale, file-based course registration application based on relational database logic.

Functional Requirements:

1. Each course in the opened courses file contains a unique course code, name, total credits, and quota information. The data to be read from this file should be saved in the ONE-WAY LINK LIST in the order indicated by the course code, as illustrated in Figure-1. Each linked list node should provide the course's code, name, capacity, and credit. Furthermore, the number of students registered in that course at each node should be kept in a dynamic array in ascending order according to the student number.
2. The linked list shown in Figure-1 should be used to open and close new courses on courses. Following completion of these actions, the data in the linked list should be stored in a file named lessons using the Save to File function.
3. This process should not be permitted if the course with the same code has already been opened. It is not possible to remove a previously opened course after the code of the course to be closed is obtained from the user. A warning should be issued. Article 4 explains the status information for students enrolled in a closed course. It should be changed to COURSE CLOSED in the StudentCourseRegistration file.

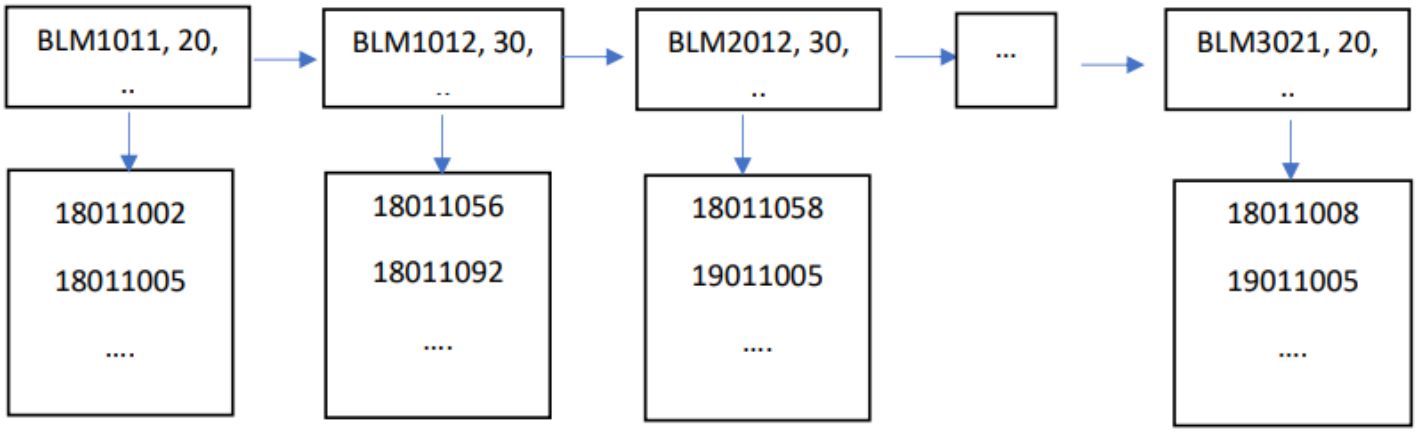

Figure-1

4. Each student's file must have their unique student number, name, surname, total number of courses enrolled, and total credits enrolled. Create a struct variable that fulfills this information and save student information in the TWO-WAY LINKED LIST (see Figure 2) named StudentList in ascending order by number. Students must be added and deleted in this linked list first, and then the data must be saved in the appropriate file.

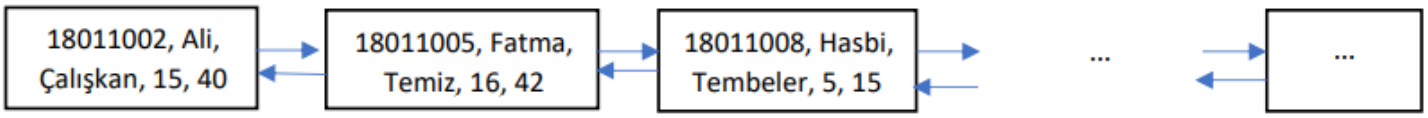

Figure-2

5. Because a student can take numerous courses, but several students can take the same course at the same time, which course has been taken by which student should be maintained in a separate file called StudentCourseRegistration, and the matching method should be <studentNO-CourseCode>. In this file, there is also a unique ID for each match that rises automatically starting at 1. The registration date and the student's registration status (REGISTERED, COURSE CLOSED, LEFT) should be preserved as supplementary information in this matching. ID, COURSE CODE, STUDENT NO, DATE, STATUS are some examples of file formats.
6. Students should be able to add courses while registering for the course based on the maximum number of courses to be registered for and the maximum credit value criteria received from the user during course opening. When these values are reached, the course-adding process should be halted. Aside from that, if a course's quota is full, the student should be prevented from enrolling in the course.
7. When a student(s) registers for or cancels a course, the addition and deletion operations on the linked list shown in Figure-1 should be performed first. The bidirectional link list in Figure-2 should then be updated as well. Following the completion of this process(es), the content of the file named StudentCourseRegistration should be updated.
8. Write a function that lists all students enrolled in a course whose course code is given. With the help of this function, print the class list of all courses to the file in COURSECODE.txt name format.
9. Create a function that lists all of the courses taken by a student with the given student number and saves it to the file RENCINO DERSPROGRAMI.txt.

Coding Requirements:

- Create a menu based on the items listed above, as well as the functions required for each operation. To avoid duplication in functions that perform similar logic, use function pointers.
- You need to define a function for all add, delete and update operations.
- You need to define a function for all add, delete and update operations.
- The use of static and global variables in your programs is prohibited.
- Any memory allocation should be done with dynamic memory management functions.
- Make appropriate structure definitions for the object in each file detailed above.
- The files to be created must be in text file format.

Note: All assignments will be run through the similarity control program; assignments that are found to be similar will be evaluated as duplicates, and each related assignment will receive a 0 (zero) grade.