COSC 311, Project #2

Due Date: October 8, 2012

In this project you will design and implement a small database system capable of the following operations:

                1- Retrieving a record

                2- Modifying a record

                3- Adding a new record

                4- Deleting a record

The database is composed of a random access file called “STUDENTS” containing student’s academic records.  The records in the STUDENTS file are of the form:

|  |  |  |  |
| --- | --- | --- | --- |
| First name | Last name | Student ID | GPA |
| 40 | 40 | 4 | 8 |

where the integers represent the field width in terms of the number of bytes. The students first and last names are at most 20 characters (Unicode characters) long, the ID is an integer, and the GPA is of type double. There are actually two files in this project, a text file and a random access file. You will create a text file containing information for, at least, 10 students (one line per student), and this program will take that information and put it in a random access file. The information can be manipulated much faster when it is stored in the random access file. All of the operations defined below use the random access file.

Your database system will maintain an index (an ordered singly linked list) for the ID field of the records in the database so it can perform the above operations more efficiently. Write a menu driven program (or a program with a GUI interface) to implement this system.  The menu should allow a user to select one of the following:

                1- Make a random-access file

                2- Display a random-access file

                3- Build the index

                4- Display the index

                5- Retrieve a record

                6- Modify a record

                7- Add a new record

                8- Delete a record

                   9- Exit

1- Make a random-access file: You have already implemented this in project #1.

2- Display the random-access file: You have already implemented this in project #1.

3- Build the index: It asks the user to enter a database name (a random-access file).  It reads the database records sequentially one at a time, and creates an ordered singly linked list in the memory.  Every node of the linked list will contain a pair (KEY, ADDRESS) as well as a reference to a node, where KEY is the student ID field and the ADDRESS is the position of the record containing the ID in the database (first record in the database is at position zero, the second record is at position 1, and so on).

4- Display the index: It asks the user to indicate whether he/she wishes to see the entire index or part of the index.  If the entire index is desired, it displays all the keys starting with the key in the first node and ending with the key in the last node. If only part of the index is desired, it prompts the user to enter the starting key.  It searches the ordered singly linked list to find the node containing that key. If the search is successful, it displays every key beginning with the starting key and ending with the key in the last node. Otherwise it displays a message indicating the failure of the search.

5- Retrieve a record: It asks the user to enter a student ID (a key value). It searches the index (the ordered singly linked list) for the key, and if the search is successful it uses the ADDRESS associated with that key to retrieve and then display the corresponding record. If the search is unsuccessful, it prints a message indicating the failure of the search.

6- Modify a record: It asks the user to enter a student ID (a key value). It searches the index (the ordered singly linked list) for the key, and if the search is successful it uses the ADDRESS associated with that key to retrieve the corresponding record. It allows the user to modify any fields of this record, except the student ID filed, and then it writes the modified record over the original record in the database.  Note that modifying a record doesn’t require any changes to the index (the ordered singly linked list) because the user is not allowed to change the ID field.

7- Add a new record: It asks the user to enter data for the new record.  This new record will be appended to the end of the database (the random-access file).  Next, the student ID (the key value) and the position (ADDRESS) of the record just written to the end of the database must be added to the index. This means a new node containing (KEY, ADDRESS) must be added to the ordered singly linked list (the index).

8- Delete a record: It asks the user to input the student ID (the key value) of the record that needs to be deleted. It searches the index (the ordered singly linked list) for the key value, and if the search is successful it uses the ADDRESS associated with that key value to delete the corresponding record from the database (using lazy deletion). It also deletes the node containing the key value (i.e., (KEY, ADDRESS)) from the index (the ordered singly linked list).

IMPLEMENTATION REUIREMENTS: The class that implements an ordered singly linked list must be a generic class.

OPERATIONAL SPECIFICATIONS: The program should always return to the main menu after processing the selected item.  The program execution should only be halted if EXIT is selected from the main menu.

SUBMISSION:  Please submit the following in a manila envelope:

            - A hard copy of the source code (well documented).

            - A CD (or DVD) that contains your project folder, readme file, input file, and the

            random access file

            - A hard copy of the input file (create an input file containing at least 10 students)

            - A sample run which tests all the menu options

GRADING:  Documentation: 10%, Program style: 5%, Friendly interface: 10%, Program works: 75%