Project: Student Gradebook

Write a program that allows the user create gradebook for students, that includes both a grade and a student’s ID number. The program should load in a list of students from a text file, then present the user with options to add a new student, delete a student from the list, change a student’s grade, and display all students. When the program closes, all changes to the list are saved onto the text file so it can be accessed again.

**Example of output**:

|  |  |
| --- | --- |
| Welcome to Student Gradebook  1. Insert New Student.  2. Delete an existing Student.  3. Change a student's grade.  4. Display a list of all students  5. Exit and save.  Please make your Selection: 4  --------------screen clears-------------------  Displaying List of students:  Student Name: Clark Kent  Student ID #: 789  Student Grade: C  Student Name: Barry Allen  Student ID #: 456  Student Grade: B  Student Name: Bruce Wayne  Student ID #: 123  Student Grade: A  Press any key to continue . . . | Welcome to Student Gradebook  1. Insert New Student.  2. Delete an existing Student.  3. Change a student's grade.  4. Display a list of all students  5. Exit and save.  Please make your Selection: 1  --------------screen clears-------------------  Insert student info :  Enter Student's First Name:  Arthur  Enter Student's Last Name:  Curry  Enter Student's ID:  963  Enter Student's Grade:  A  Press any key to continue . . . |

Each student is held in a **Student** *struct* node, each of which are created in a **StudentList** *class* which contains a **Student** *pointer* called **head**, and a number of functions used to manipulate the list in several ways.

The **Student** *struct* contains the following data.

Student \* next;

string firstName;

string lastName;

char grade;

int id;

The **StudentList** *class* contains only a Student pointer called head, and the following functions used to manipulate a list of **Student** objects and their contents.

**StudentList();**

Constructor, assigns *nullptr* to **head** pointer.

**~StudentList();**

Destructor, walks through list and deletes all nodes when **StudentList** is destroyed so that there are no **Student** nodes left in memory when the program ends.

**void insertStudent();**

Creates a new **Student** node and prompts user to enter student’s information, then places the newly created **Student** node at the front of the list. Finally the **head** pointer is updated to point to the newly created **Student** node at the front of the list.

**void deleteStudent();**

Prompts user to enter the ID of a **Student** that is to be deleted from the list. Once the user enters a valid **ID** number, the program linearly searches list until the correct **ID** is found and this node is deleted and the two adjacent nodes are linked.

**void changeStudent();**

Prompts user to enter the ID of a **Student** whose **grade** is going to be changed. Once the user enters a valid **ID** number, the program linearly searches list until the correct **ID** is found and the **grade** variable in this node is updated.

**void displayList();**

Linearly scans **StudentList** and displays data from every **Student** node until the end of the list is reached.

**void loadFromFile();**

Creates a **dataFile** *fstream* object in order to open **StudentData.txt** for input. Checks for any errors in opening the text file. Creates a while loop that runs until the end of the file is found. With each pass a new **temp** *Student pointer* is created, data is copied from the text file to the **temp** node, and then the node is added to the front of the list.

**void saveToFile();**

Creates a **dataFile** *fstream* object in order to open **StudentData.txt** for output. Checks for any errors in opening the text file. Creates a do while loop that runs until the end of the list is found. Linearly scans **StudentList** and copies data from every **Student** node to the text file until the end of the list is reached.