2100.021 - Program 1 - Students

My Solutions

Write a program that simulates a university class with a capacity of up to **7** students. A student will be implemented via a C++ class Student, whose constructor takes **student id, name, and classification (e.g. sophomore) as parameters**. Keep the information "private" in your class and implement corresponding functionality to access it as needed for the assignment.

Once the program is started, it will print out the promt "**class>** " (> is followed by a whitespace):

```
./a.out

class>
```

You will implement the following commands:

enroll

Upon entering enroll , your program will print out "class> Enroll student: " and insert a line break. Once the student information has been entered, the prompt is repeated.

```
class> enroll
class> Enroll student:
111 Popeye Freshman
class> enroll
class> Enroll student:
123 BugsBunny Junior
class>
```

You can assume that the student name and classification are single words. You must print out an error message starting with "**Error!**" if your class is full, i.e. the user tries to add more than 7 students.

For each student, you will create an instance of your student class and store it. Students are added left to right into the next available spot based on enrollment order.

drop

Upon entering drop, your program will print out "class> Enter ID: " and insert a line break. Once the student id has been entered, the prompt is repeated.

```
class> drop
class> Enter ID:
111
class> drop
class> Enter ID:
222
Error! Invalid ID.
class>
```

Make sure to print out an error message starting with "**Error!**" if a student id is selected that does not exist in the class.

If your remove a student from the class that did not enroll last, you will need to move all following students "one spot to the left". For example:

Ihe following class, there are 4 students enrolled and 3 spaces available

Student A | Student B | Student C | Student D | | | |

Removing Student B, will cause students C and D to be moved one spot to the left each:

Student A | Student C | Student D | | | | |

roster

Upon entering roster, your program will list all the students currently enrolled in the class using the following format: "id-name-classification". Notice that there are no whitespaces.The individual students are separated by commas followed by a whitespace.

For example, if there are 3 students enrolled in the class, the roster would be in the following format

```
class> roster
111-Popeye-Freshman, 123-BugsBunny-Junior, 105-PinkPanther-Sophomore
```

If there is a single student enrolled in the class, the roster would be in the following format

```
class> roster
123-BugsBunny-Junior
```

quit

Exit the program

```
class> quit
```

Submit the following files:

- Main.cpp - your main class controling the flow of the program
- Student.h - the prototype for your student class
- Student.cpp - the implementation of your student class

Example of program execution:

```
g++ *.cpp
```

```
./a.out
```

```
class> enroll
class> Enroll student:
111 Popeye Freshman
class> enroll
class> Enroll student:
123 BugsBunny Junior
class> enroll
class> Enroll student:
105 PinkPanther Sophomore
```

```
class> enroll
class> Enroll student:
321 PapaSmurf Senior
class> enroll
class> Enroll student:
1232 MickeyMouse Freshman
class> enroll
class> Enroll student:
1432 Nemo Freshman
class> enroll
class> Enroll student:
45 Batman Junior
class>enroll
Error! Class is full.
class> roster
111-Popeye-Freshman, 123-BugsBunny-Junior, 105-PinkPanther-Sophomore, 321-PapaSmurf-Senior, 1232-MickeyMouse-Freshman , 1432-Nemo-
Freshman, 45-Batman-Junior
class> drop
class> Enter ID:
111
class> drop
class> Enter ID:
1232
class> roster
123-BugsBunny-Junior, 105-PinkPanther-Sophomore, 321-PapaSmurf-Senior, 1432-Nemo-Freshman, 45-Batman-Junior
class> drop
class> Enter ID:
222
Error! Invalid ID.
class> quit
```

Grading

Your program will be judged on the following:

- 45% - Passes I/O requirements
- 40% - Code satisfies requirements of assignment
- 15% - Professional coding style
  - 5% Adequate comments
  - 5% Modularity (small main function, separate functions, etc)
  - 5% Readability (line length, indentation, variable names)

As with all other programming assignments in this class, your program will receive a 0 if it does not compile.