COMSC-205PY FALL 2020 ASSIGNMENT 3

Due November 12th at 9:00am Eastern Time

Submission checklist

Your final submission should have the following files:

- EnhancedRoster.py
- UnorderedLinkedList.py
- OrderedLinkedList.py
- LinkedListNode.py
- Roster.py

At the top of each file, clearly state:

- Your name and the name of the assignment
- Attribution for any sources used, including people (other than instructors) who you asked for help
- Description of the file

Read all the way through this assignment twice before you start working on it. The second time through, take notes, plan out how you will work on this over the next 6 days, and work on a design for the program.

Background

For this assignment, download the `UnorderedLinkedList`, `OrderedLinkedList`, and `LinkedListNode` classes in the starting code from Moodle. These will be used to create rosters.

The Assignment

This assignment has two parts: a basic roster, and an enhanced roster. The basic version should be created first.

Basic Roster Program

Design and implement the program `Roster.py` that will handle the addition and removal of students to a class roster. You **must** use an ordered linked list as the core data structure. The program should do the following when run:

1. Ask the user to specify the maximum size for the class roster
2. Repeatedly
   a. get a command from the user
   b. if the command is legal, carry out that command using the roster. If the command is illegal, get a new command.

The user can type any of the following four commands to manage the roster:

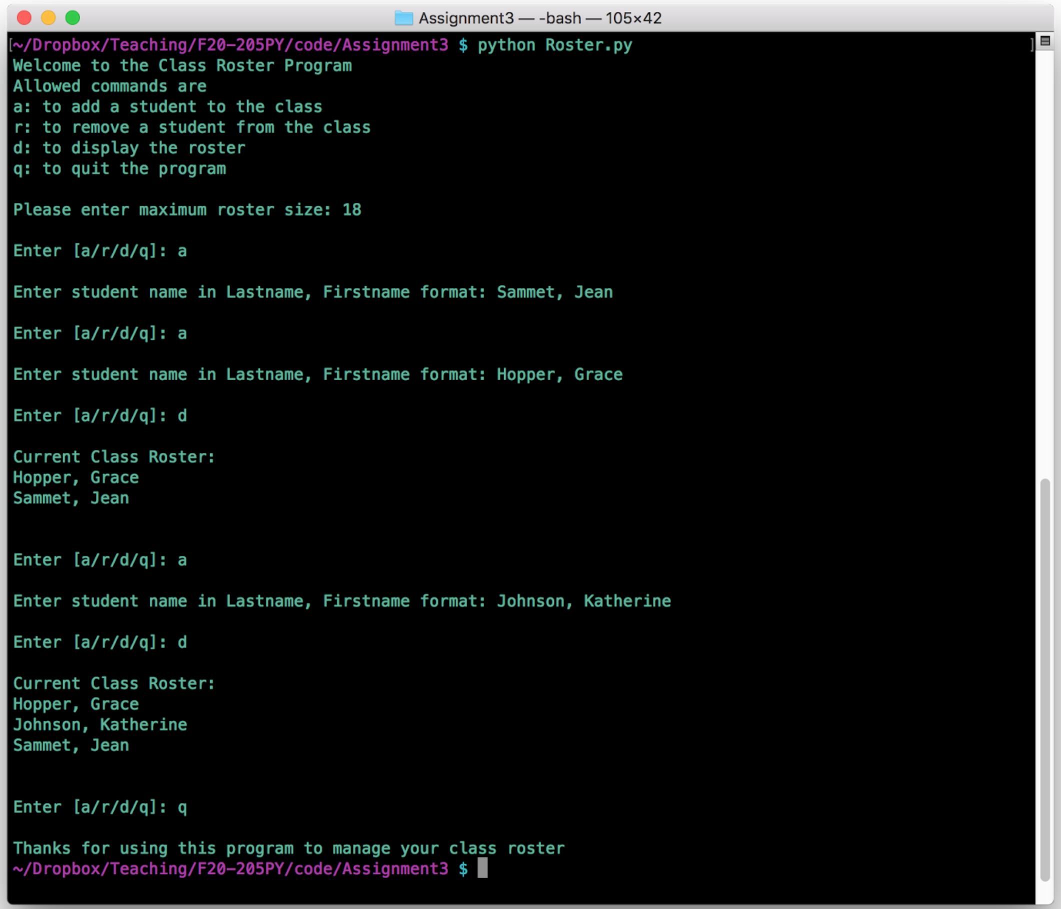
- **a** to add a student to the class; this should lead to another prompt to enter the student

name
- **r** to remove a student from the class; this should lead to another prompt to enter the student name
- **d** to display the roster
- **q** to quit the program

Information about data and program operation:

- The program expects data in **Lastname, Firstname** format (with a comma after the Lastname).
    - This should be entered all on one line, in that order, with that syntax, exactly.
    - Assume the user will always use this format; you don't need to handle bad input.
- The roster must be kept in alphabetic order by last name.
- As soon as the roster reaches the specified maximum size, a message should be printed saying the roster is full, and then print the roster and end, without any other prompts.
- All student names on the roster should be displayed, one per line, in the form Lastname, Firstname.

A sample run of the Roster program is shown below.



```
~/Dropbox/Teaching/F20-205PY/code/Assignment3 $ python Roster.py
Welcome to the Class Roster Program
Allowed commands are
a: to add a student to the class
r: to remove a student from the class
d: to display the roster
q: to quit the program

Please enter maximum roster size: 18

Enter [a/r/d/q]: a

Enter student name in Lastname, Firstname format: Sammet, Jean

Enter [a/r/d/q]: a

Enter student name in Lastname, Firstname format: Hopper, Grace

Enter [a/r/d/q]: d

Current Class Roster:
Hopper, Grace
Sammet, Jean


Enter [a/r/d/q]: a

Enter student name in Lastname, Firstname format: Johnson, Katherine

Enter [a/r/d/q]: d

Current Class Roster:
Hopper, Grace
Johnson, Katherine
Sammet, Jean


Enter [a/r/d/q]: q

Thanks for using this program to manage your class roster
~/Dropbox/Teaching/F20-205PY/code/Assignment3 $
```

Implementation note: you do not need to define classes or create objects for this assignment. A reasonable approach would be to put the code for gathering user input in a main() function, and defining other helper functions as you see fit. For example, you might make an addStudent function that gets called when the user command is **a** (for add).

Enhanced Roster Program

Make a copy of Roster.py. Call the copy EnhancedRoster.py. Then make two sets of modifications as follows:

Enhanced user protections

Roster.py is pretty basic and has no error checking. It does not guard against the user trying to do bad things with the roster. First, in real usage it is possible that the user might try to remove a name that is not on the roster. We would like the program to behave in a reasonable fashion -- but the OrderedList starter code assumes that we only try to remove items that are on the list, and will crash if you try to remove an item that is not on the list. Add functionality in Enhanced Roster so that it gives helpful messages (and prevents crashes) as follows -- you do NOT need to modify OrderedLinkedList:

- user tries to remove a name that is not on the roster
- user tries to add a name to the roster that is already there -- a name should not be on the roster twice
- user tries to remove a name from an empty roster -- the user should not be asked for a name if the roster is empty

Enhanced operation

Enhance your program with the following added functionality:

- Once the roster is full (holding the specified maximum number of students), instead of the program ending, any student added subsequently will be added to a second list, the “wait list”. The wait list should **not** be kept in alphabetic order (use UnorderedLinkedList for this).
- When a student is removed from a full roster, if there are any students on the waitlist then the first student on the waitlist should automatically be added to the roster, placed in correct alphabetical order, and removed from the waitlist.
- The display command should print both the roster and the waitlist.
- The waitlist should have the same protections and functionalities as the roster: duplicate names should not be allowed, names should be removable with the r command, etc.

Rubric

Each section will be given a letter grade. The average of those letters will generate your final score. Note: Moodle does not allow letters, so they will be translated into a percentage there.

	F	D	C	B	A
Input	No inputs possible	Only one command is asked for, or only one input works	Limited commands are asked for, or only two inputs work	Commands are asked for until program ends, but one input doesn't work, or the inputs are in a different format than specified	Commands are asked for until program ends, and all inputs work as expected
Basic Operation	No Roster	Three or four of the requirements are missing from A.	Two of the requirements from A are missing, or most of the requirements work with bugs.	One of the requirements from A is missing, or two requirements work with bugs	Names can be added. Names are added in alphabetical order by last name, and accepts the input in "Lastname, Firstname" format. Names can be displayed. Names can be removed. When the list is full, the program quits.
Enhanced User Protections	No Enhanced Roster	None of the protections from A work, but Enhanced Roster exists	Two of the protections from A do not work	One of the protections from A does not work, or changes were made to Roster instead of EnhancedRoster	The following things are prevented: removing a name from an empty list, removing a name that doesn't exist, adding a duplicate name to the list
Enhanced Operation	No Enhanced Roster	One or two of the operations from A work	Three or four of the operations from A are not present	One or two of the operations from A are not present, or changes were made to Roster instead of EnhancedRoster	Waitlist exists, and is added to when roster is full. Waitlist is ordered based on when added, NOT alphabetical. When a student is removed from the list, the next student is added from the waitlist, in order. Display prints both the list and waitlist. Waitlist has the same functionality as the roster. The program does not quit when the list is full.

Comments, naming, style	No comments present, no name in file, method names incorrect	Name in file, some methods named correctly, no comments	Name in file, description of file in comment, methods named correctly (typos ok), more than one error that prevents running	Name in file, description of file and most methods, methods named correctly, some variables named meaningfully, significant over- or under-use of comments or one error that prevents running	Name in file, description of file and all methods, methods named correctly, all variables named meaningfully, other comments as needed, no errors that prevent running

Have you read all the way through this assignment twice? If not, read it again!

Do you have questions? Are there things you don't understand? Talk to classmates and ask questions on Piazza or discord.