COSC-211: Data Structures
HW0: Getting Started

Due Thursday, September 8, 11:59pm ET

There are two purposes of this assignment. The first is to give you a programming refresher involving some exercises that you should be able to do without difficulty having completed COSC-112. The second is to familiarize you with the Gradescope submission system that we'll be using this semester. We'll be doing both programming work and written work this semester, so, in addition to the programming part of this assignment, there is also a short written component to give you practice submitting written assignments.

1 Your Tasks

Go to www.gradescope.com, click "log in," and choose to log in with school credentials. Enter your Amherst username and password. I've automatically enrolled you in COSC-211 on Gradescope, so you should see the course when you log in. If you do not see the course, please email me right away.

1.1 A short written assignment

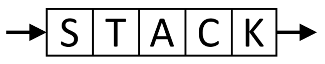
This is a very short written assignment to remind you of some of the data structures you've seen in the past. There are three short problem, and then we'll walk you through submitting your work to Gradescope. You should write up this assignment however you plan to write up your written work throughout the semester: if you think you'll hand-write your work, hand-write this assignment, and if you think you'll type up your work, type up this assignment. If you choose to hand-write you work, please make sure your writing is neat and legible. Our graders reserve the right not to grade your work if they can't read what you've written.

Problem 1: Draw a picture of a singly linked list that contains the letters in your first name. The list should be ordered so that we read your name when traversing the list from the head to the end of the list.

Problem 2: Suppose you have an initially empty stack. What does the stack contain after the following sequence of operations?

```
push(2)
push(1)
push(3)
push(6)
pop()
pop()
push(1)
push(5)
pop()
```

Problem 3: Here's a picture of a queue with some elements in it:



Draw a picture of what this queue will look like after you add the elements `Q`, `U`, `E`, `U`, `E`, in that order.

Now you're ready to submit your work to Gradescope.

Before submitting your work, please scan it as a pdf (if you wrote any part of your solutions by hand) or save it as a pdf (if you typed your solutions). You may need to download a phone app to be able to scan to pdf; one option is the Dropbox app: https://www.dropbox.com/features/productivity/doc-scanner-app.

Go to the COSC-211 course on Gradescope, select the assignment you're submitting (in this case, that assignment is "HW0: Written"), then select "upload pdf" to upload the pdf scan of your submission. For each question, select the pages of your scan that correspond to that question. This step is very important! If you don't indicate which pages belong to which problems, we will assume that the problem was not submitted and your work won't be graded. Click here for a video with instructions on how to do this.

When you're done selecting the pages for each problem, click save or submit.

1.2 A short programming assignment

The next part of the assignment is a short programming assignment. In addition to giving you practice submitting code to Gradescope, this portion of the assignment is intended to help you (and me) assess the extent to which you've mastered the prerequisite material for this course. **If you find this assignment challenging, please come talk to me.** This is an indicator that the course is likely to be difficult, and it's important that we come up with a plan early on to make sure you're on track.

1.2.1 Programming tasks

In this part of the assignment, you will implement a Linked List that stores data of type `String`. You will then write some methods to perform various operations on the Linked List. You are welcome to use any platform and any IDE that you like, as long as we can successfully compile and run your code on the submission platform (see the next section below for how to check this).

Your tasks are as follows:

1. Create two new java classes called `LinkedList` and `Node`. Your code **must** be saved in two files called `LinkedList.java` and `Node.java` respectively (exactly like that, caps included). It is very important to follow any naming conventions we give you for files,

methods, etc. If you don't, your program won't run properly with our test code and we won't be able to grade your work.

2. Use your `LinkedList` and `Node` classes to implement a linked list that stores `Strings`. This should be a doubly linked list with a header pointer.

3. Your implementation should include the following methods, with headers exactly as they appear below (these are listed in the order in which I suggest implementing them):

   - `public void add(String s)`
     Adds the data item `s` to the end of the list.
   - `public void traverse()`
     Traverses the list, printing out all data items in order from the header to the footer.
   - `public int count(String s)`
     Counts the number of times that data element `s` appears in the list, and returns that value.
   - `public void add(String s, int p)`
     Adds the data item `s` to the list at position `p` (where the first data element is at position 0).
   - `public String remove(int p)`
     Removes and returns the data element at position `p`. Returns `null` if `p` is out of bounds.
   - `public boolean remove(String s)`
     Removes the first occurrence of `s`, if `s` appears in the list. Returns `true` if `s` did appear, and `false` if not.
   - `public boolean removeAll(String s)`
     Removes all occurrences of `s`, if `s` appears in the list. Returns `true` if `s` did appear, and `false` if not.
   - `public boolean swap(int p, int q)`
     Swaps the elements at positions `p` and `q`. Returns `true` if the operation was completed successfully, and `false` if not (because `p` or `q` was out of bounds). Note that this can be accomplished easily by swapping the data element stored in the node at position `p` with the data element stored in the node at position `q`. **Do not implement your method this way**; the goal here is for you to practice manipulating pointers, so you should implement this method by swapping the positions of the nodes themselves.
   - `public void reverse()`
     Reverses the order of the list. Again, do this *without* changing the node in which any data element is stored.

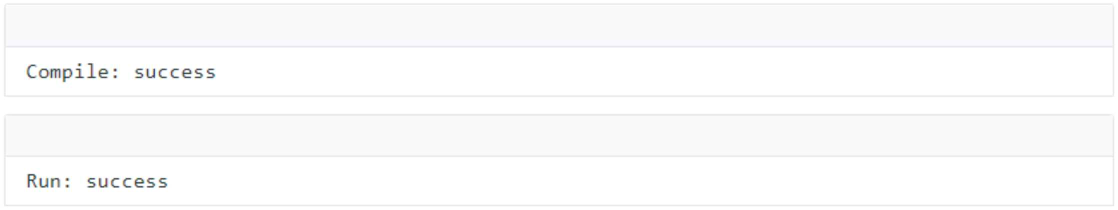
4. Write some code, in a separate file called `Tester.java`, to test that your Linked List works properly. I strongly recommend testing each method individually after you write it, before moving on to the next. In a few weeks, we will discuss how to approach code testing. For now, your goal is to convince yourself that your code works properly before submitting.

1.2.2 Submission

Once you've written your program, you're ready to submit. Go to the COSC-211 course on Gradescope, and select the "HW0: Programming" assignment. Upload your `LinkedList.java`, `Node.java`, and `Test.java` files.

At this point, our autograder will run your program. We'll be using Gradescope's autograder throughout the semester to check that your code compiles and runs without errors in the environment in which we'll be grading it. You're welcome to use any platform and any IDE that you like, as long as your code passes the autograder's compile/run tests. For this assignment, all the autograder does is check that your code compiles and runs without errors. Later, we might also include some simple test cases that you can use as a starting point to verify that your code does what it's supposed to.

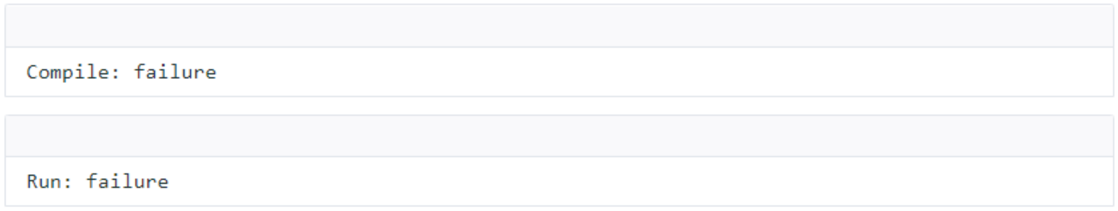
If your code successfully compiled and ran, you'll see a message on Gradescope that looks something like this:



```
Compile: success
```

```
Run: success
```

If you see the above message, you're all set! Your code has been submitted, runs on our system, and is ready for us to grade. If your code had errors when compiling or running, you'll instead see a message that looks something like this:



```
Compile: failure
```

```
Run: failure
```

This message means that you have more work to do! You should fix whatever compiler or runtime error occurred, and then resubmit your code. (Gradescope won't tell you what the specific error was - that's your job to figure out! If your code compiles on your machine but not through the autograder, check if your IDE added, e.g., a line that begins "package" at the start of your code. Some IDEs, such as Eclipse, will do this, and you'll need to delete this line before submitting your code.) You can resubmit as many times as you want before the submission deadline.

2 Submit your work

The above sections walk you through submitting this assignment on Gradescope.

This assignment is due Thursday, September 8, 11:59pm ET.