CSCI 1933 Lab 8

Lists

Lab Rules

You may work individually or with *one* partner in lab. We suggest that you have a TA evaluate your progress on each milestone before you move onto the next. The labs are designed to be completable by the end of lab. If you are unable to complete all of the milestones by the end of lab, you have **until the last office hours on Monday** to get those checked off by **any** of the course TAs. We suggest you get your milestones checked off as soon as you complete them since Monday office hours tend to become crowded. If you worked with a partner, both of you must be present when getting checked off to receive credit. You will only receive credit for the milestones you have checked off by a TA. There is nothing to submit to Moodle for this lab.

Attendance

Per the syllabus, students with more than 3 unexcused lab absences over the course of the semester will automatically fail the course. The TAs will take attendance during lab; it is expected that students will be actively working on the lab material. *Your physical presence in lab is not sufficient to be marked as present for the purposes of attendance.*

1 List Slicing

List slicing is a way to acquire specific subsections of a list. If you've used Python before, you might remember that Python's lists have a built in slicing method. Download the completed `AList.java` from last week's lab, and define the following methods:

- `public AList<T> slice(int start, int stop)` – This should return an `AList<T>` which contains the elements from the start, inclusive, to the end, exclusive. See the example below for more details.
- `public AList<T> slice(int start, int stop, int step)` – The functionality for this method should be exactly the same as the method above, except that this method should only choose elements every `step` away.

Given the following `AList<String> l` as `[a, b, c, d, e, f, g, h]`, then `l.slice(1,7,2)` should return the `AList<String>` containing `[b, d, f]`.

***Important*:** Your methods should make **no** changes to the original `AList`. Instead, they should each return a new `AList` containing the proper elements.

Milestone 1: Write tests for each of your `slice` methods, and show them to your TA.

2 Sorting

So what's going on with our very first line for `AList`? Well, one of the advantages to Generics is the ability to only allow specific objects to be used as Generics. In this case, we only allow Generics which `extend Comparable`. This means we can use any of the `Comparable` methods in our code!

For the second milestone, create a method `public void sort(boolean ascending)` which sorts your array in either ascending or descending order, depending on the value of `ascending`. You have access to the `compareTo()` method, so it might be wise to use it!

Milestone 2: Create a few tests for your `sort()` method, then show them to your TA.

3 File Input

We've worked a bit with file input and output in previous labs, but to bring everyone back up to speed for the current project, you will be reading input from a file and placing the contents into a new `AList`. Create the method `public static AList<String> fileToAList(File input)`. Each line of the input file will contain an index and a string in the format below. Your task is to place each string in an AList at the proper index.

All files will be input in this format: `[index], [string]`. An input of this

```
1, CSCI
2, 1933
0, hello
```

Should return an AList (after toString()) like this: `[hello, CSCI, 1933]`

Hint: Your `AList` will very likely not have the proper number of entries to immediately put every line into its proper place. How can you get around this issue?

Milestone 3: To test your function, download the provided `input.txt` file from moodle and turn it into an AList. In addition, think of the answers to the following questions: Is this the best way to represent this file input? Is there a way to make this easier? Show your TA the output, your code, and your wisdom.