Assignment 5

Recursion

Introduction

In this assignment, you will use recursion to solve a number of problems. In Part 1, you will get some experience writing recursive algorithms that operate on a linked list of songs, and in Part 2 you will solve problems recursively working with a generic implementation of the List ADT (that for the last few exercises is working with Integers objects specifically).

The automated grading of your assignment will include some different and additional tests to those found in the `A5Tester.java` file, as it does not include a comprehensive set of sets for each method. You are expected to write additional tests until you are convinced each method has full test coverage. The displayResults and test coverage videos provide more information about code testing.

Objectives

Upon finishing this assignment, you should be able to:

- Write recursive methods that operate on a linked list
- Write recursive methods that operate on an implementation of the List ADT
- Use a context-preserving accumulator in a recursive solution

Submission and Grading

Attach `LinkedSongList.java` and `A5Exercises.java` to the BrightSpace assignment page. Remember to click **submit** afterward. You should receive a notification that your assignment was successfully submitted.

If you chose not to complete some of the methods required, you **must** provide a stub for the incomplete method(s) in order for our tester to compile. If you submit files that do not compile with our tester, you will receive a zero grade for the assignment. It is your responsibility to ensure you follow the specification and submit the correct files. Additionally, your code must not be written to specifically pass the test cases in the tester, instead, it must work on all valid inputs. We may change the input values during grading and we will inspect your code for hard-coded solutions. This video explains stubs.

Be sure you submit your assignment, not just save a draft. All late and incorrect submissions will be given a zero grade. A reminder that it is OK to talk about your assignment with your classmates, but not to share code electronically or visually (on a display screen or paper). Plagiarism detection software will be run on all submissions.

Instructions

For this assignment, it may be a good idea to jump back and forth between Part 1 and Part 2, as the exercises found in both parts get progressively more difficult.

For each of the exercises in both parts of this assignment, you will need to call a recursive helper method in order to recursively operate over the elements in the collection. It might be a good idea to reference the lecture examples if you forget how to setup and call a recursive helper method.

Part 1:

1. Download all of the `.java` files found in the *Assignments* > *Assignment 5* page on BrightSpace.
2. Read through the tests provided in `A5Tester.java`. You will be implementing methods within the `LinkedSongList.java` file.
3. Compile and run `A5Tester.java`. Work through implementing each of the incomplete methods found in the `LinkedSongList` class one at a time. Debug each method until all of the tests pass for that method before proceeding to the next method.
4. **Remember:** You must solve each problem recursively. There must not be any `for` or `while` loops in your solution, or you will receive a grade of 0.

Part 2:

1. For Part 2, you will be implementing methods in the `A5Exericses.java` file.
2. Read the documentation provided in `List.java`. When using a list to solve a problem, you can only use the methods specified in the `List` interface. Think about which methods you will need to use to implement all of the important steps in a recursive algorithm.
3. Compile and run `A5Tester.java`. Work through implementing each list method found in Part 2 of the `A5Exercises.java` file one at a time. Debug the method until all of the tests pass for that method before proceeding to the next method. You should write tests to ensure your solution works for all valid input scenarios.
4. **Remember:** You must solve each problem recursively. There must not be any `for` or `while` loops in your solution, or you will receive a grade of 0.

CRITICAL: Any compile or runtime errors will result in a **zero grade** (if the tester crashes it will not be able to award you any points for any previous tests that may have passed). Make sure to compile and run your program before submitting it!