

Analysis of Algorithms
CS 375, Spring 2020

Homework 3

Due **AT THE BEGINNING OF CLASS** Wednesday, February 19

- Unless otherwise specified, exercises will be from the CLRS textbook and will be named on HW assignments by exercise number used in the book.
- A few essential style guidelines for writing algorithms in CS375:
 - Unless explicitly instructed otherwise, avoid using **break** statements (or **continue** or **pass**) statements in algorithms for CS375—employing them can interfere with common approaches to reasoning about code correctness (e.g., using loop invariants). Instead, please find other approaches, perhaps using boolean variables to indicate loop termination conditions.

Answers employing **break** (**continue**, **pass**) may not receive full credit. Please feel free to talk with me about any particular situations that might come up—I'm happy to talk with you about how to avoid **breaks** in your code!
 - When presenting an algorithm, avoid using language-specific commands/routines (without comments) in pseudocode. Pseudocode, by definition, is supposed to be language-independent, at a level of abstraction higher-level than particular programming languages.
 - Please be sure to initialize variables or declare inputs/outputs, and be sure the purpose of **every variable**—including the names of functions or algorithms—can be quickly and fully understood by a reader; typically, this is done either by using descriptive variable names or by describing variables in English. Algorithms in which variables cannot be promptly understood may not receive full credit, so please feel free to ask me about variable names and descriptions—I'm happy to talk with you about particular situations that might come up!
 - Often, the clearest way to present an algorithm is with both pseudocode and English description. Pseudocode must be accompanied by some kind of clarifying English description to count as fully explained. (And see below about the importance of explanations on CS375 HW!)

Please note that, unless explicitly instructed otherwise, these guidelines apply to all exercises in CS375, whether or not they explicitly appear as part of the assignment itself. (This is similar to how all answers should be accompanied by explanations unless explicitly specified otherwise, whether or not that instruction is explicitly given as part of the assignment.)

- *A general note for CS375:* When writing up your homework, please write neatly and **explain your answers clearly**, giving all details needed to make your answers easy to understand. Graders may not award full credit to incomplete or illegible solutions. Clear communication *is* the point, on every assignment.

In general in CS375, unless explicitly specified otherwise, answers should be accompanied by explanations. Answers without explanations may not receive full credit. Please feel free to ask me any questions about explanations that might come up!

Exercises

1. Using the *LList* data structure from class, write a recursive algorithm for the *reverse* problem on lists:

```
# Input: List  $L = [a_0, a_1, \dots, a_n]$ 
# Output: List  $L' = [a_n, \dots, a_1, a_0]$  with the same elements as in  $L$ 
#           but in reverse order
```

You may not use other data structures in your algorithm—this is an exercise on LLists! As usual, give a short English explanation of correctness; because the algorithm is recursive, make sure it's an inductive explanation.

Note: You are welcome to create new functions for use on LLists to use as part of your solution! For each such function, however, you must explicitly write the algorithm (using the LList functions given in the definition from class) as part of this exercise, and you may not access the parts / fields of LLists in any way other than those provided by the definition from class.

As usual, give a short English explanation of correctness for every algorithm given for this exercise—at least *LLReverse*, but also any other algorithms you create for use here. For each recursive algorithm on LLists (and I anticipate any algorithm you write for this exercise would be recursive!), make sure the accompanying explanation is an inductive explanation.

2. **(Same problem, new syntax!)** Using Python-like list-slicing and indexing syntax in pseudocode, along with Python-like list functions `append` and `extend` (if needed), write a recursive algorithm for the *reverse* problem on lists.

As usual, give a short English explanation of correctness. (You can certainly refer to your explanation from the previous exercise when doing this one! Because the operations used here are different, though, the explanation here will not be completely identical to the previous one.)