Exercise 3: Stacks and Chains of People

Due date: Sunday October 8, before 10pm

Starter code

- ex3.py **[submit]**
- stack.py (*updated* October 4, removes the "property" part)
- ex3_test.py

Task 1: More Stack Exercises

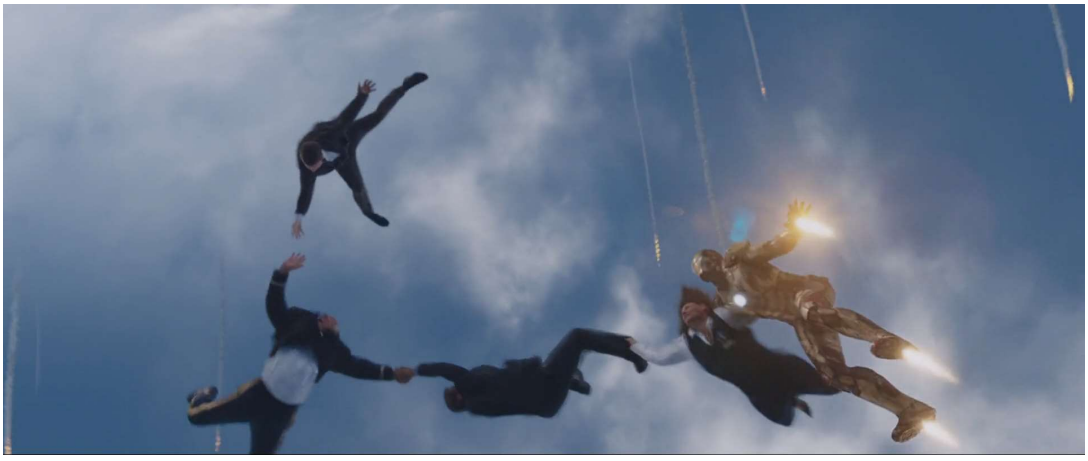
In this exercise, you'll practice using stacks. Once again, your code should only use the four methods defined in the stack public interface, and not use anything else about a stack implementation. **If you try to access the `_items` attribute, you will likely get a 0 on this part of the exercise.**

In the starter code, your first task is to implement two functions that operate on Stacks. Do *not* use any built-in compound data types like Python lists or tuples; instead, if you need to use one or more temporary `Stack` objects to store multiple necessary values.

If you get stuck, wait until after Wednesday's lecture and review the work we've done with Stacks so far! You'll probably find some inspiration. ;)

Task 2: A Chain of People

In this task, you'll begin thinking about a new way to model lists, using *links between nodes*. We'll start with a "concrete" example.



Consider the situation illustrated in the picture above: some people are forming a human chain. This chain has a *leader*, who is holding onto someone, who is holding onto someone else, etc. Note that the last person in the chain isn't holding onto anyone. For simplicity, we store only two pieces of information about each person: their name, and the "next" person in the chain, i.e., who they are holding onto.

We've provided a simple model of this situation in the starter code, consisting of two classes: `PeopleChain` and `Person`. The initializer is provided for you—try reading the code if you'd like, but this

isn't necessary for completing this task. We'll talk about the details next week!

Implement the `PeopleChain` methods `get_leader`, `get_second`, `get_third`, and `get_nth`. Note that `get_nth` generalizes the other three methods, meaning all three of them could be implemented with `get_nth`. You can do this if you want, but we *strongly recommend* that you implement the methods in the order provided, to get a good feeling of what's going on.

Finally, one implementation restriction is that **you must implement `get_nth` with a `for` or `while` loop**. In particular, if you know what recursion is already, *don't use it here*! (You'll use recursion to your heart's content soon enough, my friend.)

Hint: if you need help, think about looping through the first `n` people in the chain. Have a variable store a reference to the "current person"; how would you advance from the current person to the next person? Take a look at the `PeopleChain` initializer for a bigger hint.

Do not use `eval` for this (or any other) exercise.

Submission instructions

Submit your `ex3.py` file on MarkUs. You should *not* need to submit any other files (e.g., the testing file or stack.py). For more detailed instructions, please consult our Exercise policies page.

For course-related questions, please contact **csc14817f@cs.toronto.edu**.