Lab #10
Recursion

Z. Hutchinson
zachary.s.hutchinson@maine.edu

November 16, 2021

Goals

The goal of this lab is to practice using Python 3. Specifically, you will practice:

- Recursion
- List algorithms

Instructions

All work is due at the **end of your lab** and must be submitted to Brightspace in the proper place. Unless otherwise instructed, submissions must be python files (e.g. files that end with *.py*). Any other format, even if it is plain text, will **not** be graded. Messy or otherwise unreadable code will lose points. Lab submissions can be all in the same file, but please label with comments to which task code belongs. IMPORTANT: Any code that is commented out will not be graded. **RUN YOUR CODE TO MAKE SURE IT WORKS!!!**

Task 1

Write a recursive summation function called *recsum*. *recsum* has one argument, *N*, of type integer. It returns the sum of all the numbers between *N* and 1, inclusive. The function must calculate the sum recursively.

Task 2

Write a function that determines (recursively) how many 4 feet sections a board can be cut into. The recursive function takes one parameter, the current board length. The function should return the number of 4 feet lengths. Consider carefully, how you will reduce the problem with each recursive call, what the base case is and what should be returned.

Task 3

Write a recursive reversing function. The function takes a list of items as a parameter. If the length of the list is 1, the list is returned. If the length is greater than 1, it divides the list in half and calls itself on each half. The return values from each half are swapped into a new list. For example, the return value from the recursive call containing the first half of the list will be concatenated second to the new list. The return from the second half will be concatenated first.

Task 4

Implement the modern Fisher-Yates list shuffle algorithm. Fisher-Yates goes like this:

```
A is a list of N elements.
for i from 0 to N-2 (where N is the size of the list)
    pick a random j, such that i <= j < N
    swap A[i] and A[j]
```

Test the algorithm.