Homework Assignment 3

Due Wed May 1 11:59pm

You may work with a partner on this assignment but both of you must submit your own solutions to gradescope. It is important to note your partners name under yours in the gradescope submission.

Recursion

Problem 1 (20 pts)

Write a recursive Python function findSmallest that expects one argument, a Python list of integers, and returns the smallest integer in the list. Thinking recursively, the smallest integer is either the first integer in the list or the smallest integer in the rest of the list, whichever is smaller. If the list has only one integer, then the smallest integer is this single value. You may assume that the list has at least one element.

Here are some examples:

```
>>> findSmallest([1, 8, 34, 12, 0, 99, 10])
0
>>> findSmallest([42])
42
```

Helpful Python syntax: If A is a list of integers, and you want to set the list B to all of the integers in A except the first one, you can write B = A[1:]

Problem 2 (20 pts)

Use Python and recursion to write a function called findValue that determines whether a given data value is in a linked list. The function returns True if the value is present and False otherwise. Do not use the search function.

Your recursive function should take two arguments. The value being searched and the head of a non-empty linked list of Nodes.

Here are some examples:

```
n1 = Node(10)
n2 = Node(20)
```

```
n3 = Node(30)
head = Node(40)
n2.setNext(n1)
n3.setNext(n2)
head.setNext(n3)


>>> findValue(10,head)
True
>>> findValue(40,head)
True
>>> findValue(50,head)
False
>>>
```

Note that your function must work on lists other than the example above. You may assume that the list has at least one element

Problem 3 (25 pts)

Assume you are climbing a ladder. The ladder has n rungs. You can only climb one or two rungs at a time. This prompts you to wonder how many different ways there are to climb to the top. For example, if there are three rungs on the ladder, there are three different ways to climb to the top:

1 rung + 1 rung + 1 rung
1 rung + 2 rungs
2 rungs + 1 rung

For the general question, use Python and recursion to write a function called ladder that calculates the number of different ways you can get to the top.

Here are some examples:

```
>>> ladder(3)
3
>>> ladder(5)
8
>>> ladder(10)
89
```

Problem 4 (25 pts)

Write a recursive function recPal(word) to check if a string is a palindrome. You will probably want to use the string indexing and slicing featured in Python to obtain substrings or you could use a Deque as shown in class.

Problem 5 (10 pts)

In class, and book chapter 4.12 (ActiveCode 1), we show how memoization (or caching) can be used to speed up recursive procedures that are burdened by many redundant recursive calls. Caching works by only making recursive calls the first time, subsequent calls are retrieved from the cache.

In this problem you will memoize your recursive solution to ladder() so that it will work for large numbers, i.e. long ladders. Do this by adding a Python dictionary called knownResults as the second parameter to ladder. It will store all results that have been previously computed. Always check knownResults before making a recursive call, and store results from recursive calls into knownResults. Your new definition line will look something like.

```
def ladder(length, knownResults = {}):
```

The calls to ladder outside the function will still look like:

```
>>> ladder(10)
89
```