CS 61A Lists, Recursion, Tree Recursion
Spring 2020 Guerrilla Section 1: February 22, 2020 Solutions

1 Lists, Recursion, Tree Recursion

Questions

1.1 What would Python display?

```
lst = [1, 2, 3, 4, 5]
lst[1:3]
```

```
[2, 3]
```

```
lst[0:len(lst)]
```

```
[1, 2, 3, 4, 5]
```

```
lst[-4:]
```

```
[2, 3, 4, 5]
```

```
lst[3:]
```

```
[4, 5]
```

```
lst[1:4:2]
```

```
[2, 4]
```

```
lst[:4:2]
```

```
[1, 3]
```

```
lst[1::2]
```

```
[2, 4]
```

```
lst[::-1]
```

```
[5, 4, 3, 2, 1]
```

```
lst + 100
```

```
Error (These aren't numpy arrays)
```

```
lst3 = [[1], [2], [3]]
lst + lst3
```

```
[1, 2, 3, 4, 5, [1], [2] , [3]]
```

1.2 Draw the environment diagram that results from running the code below

```
def reverse(lst):
    if len(lst) <= 1:
        return lst
    return reverse(lst[1:]) + [lst[0]]

lst = [1, [2, 3], 4]
rev = reverse(lst)
```

https://goo.gl/6vPeX9

1.3 Implement a function *map_mut* that takes a list as an argument and maps a function f onto each element of the list. You should mutate the original lists, without creating any new lists. Do NOT return anything.

```
def map_mut(f, L):
    >>> L = [1, 2, 3, 4]
    >>> map_mut(lambda x: x**2, L)
    >>> L
    [1, 4, 9, 16]
```

```
def map_mut(f, L):
    for i in range(len(L)):
        L[i] = f(L[i])
```

1.4 Check your understanding

1 When copying the list, when are you copying a pointer of the list vs. copying the actual value inside of a list?

We copy pointers when we refer to a list within a list or another object and we copy the actual values of the list when the item inside that box is a primitive. We also copy pointers of the entire list when we assign variables to that list because we copy pointers for objects.

2 How would you make a deep copy of a list?

Recurse through the list and for every element in the list that is also a list object, copy, the elements of the list object into your copy. In other words, we create a function deep_copy(lst) which takes in a list. We go through each element of that lst, and if the element is type list, we call deep_copy on that sublist. We eventually go through the entire lst that we pass in, and return our deep copy list back. OR: Import copy, copy.deepcopy()

1.5 What are three things you find in every recursive function?

```
1) Base Case(s)
2) Way(s) to reduce the problem into a smaller problem of the same type
3) Recursive case(s) that uses the solution of the smaller problem to solve the original (large) problem
```

1.6 When you write a Recursive function, you seem to call it before it has been fully defined. Why doesn't this break the Python interpreter?

When you define a function, Python does not evaluate the body of the function.

1.7 Below is a Python function that computes the nth Fibonacci number. Identify the three things it contains as a recursive function (from 1.1).

```python
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```

```
Domain is integers, range is integers.
Base Cases: if n == 0: ..., elif n == 1: ...
Finding Smaller Problems: finding fib(n - 1), fib(n - 2)
Recursive Case: when n is neither 0 nor 1, add together the fib(n - 1) and fib(n - 2) to find fib(n)
```

1.8 With the definition of the Fibonacci function above, draw out a diagram of the recursive calls made when **fib(4)** is called.

```
                  fib(4)
                  /    \
              fib(3)   fib(2)
             /    |      |    \
         fib(2) fib(1) fib(1) fib(0)
        /    |
    fib(1)  fib(0)
```

1.9 What does the following function **cascade2** do? What is its domain and range?

```
def cascade2(n):
    print(n)
    if n >= 10:
        cascade2(n//10)
        print(n)
```

Domain is integers, range is None. It takes in a number n and prints out n, then prints out n excluding the ones digit, then prints n excluding the hundreds digit, and so on, then back up to the full number.

1.10 Consider an insect in an `M` by `N` grid. The insect starts at the bottom left corner, `(0, 0)`, and wants to end up at the top right corner `(M-1, N-1)`. The insect is only capable of moving right or up. Write a function `paths` that takes a grid length and width and returns the number of different paths the insect can take from the start to the goal. (There is a closed-form solution to this problem, but try to answer it procedurally using recursion.)

```
def paths(m, n):
    """
    >>> paths(2, 2)
    2
    >>> paths(117, 1)
    1
    """

    if m == 1 or n == 1:
        return 1
    return paths(m - 1, n) + paths(m, n - 1)
```

1.11 Write a procedure `merge(s1, s2)` which takes two sorted (smallest value first) lists and returns a single list with all of the elements of the two lists, in ascending order. Use recursion.

Hint: If you can figure out which list has the smallest element out of both, then we know that the resulting merged list will have that smallest element, followed by the merge of the two lists with the smallest item removed. Don't forget to handle the case where one list is empty!

```
def merge(s1, s2):
    """ Merges two sorted lists
    >>> merge([1, 3], [2, 4])
    [1, 2, 3, 4]
    >>> merge([1, 2], [])
    [1, 2]
    """


    if len(s1) == 0:
        return s2
    elif len(s2) == 0:
        return s1
    elif s1[0] < s2[0]:
        return [s1[0]] + merge(s1[1:], s2)
    else:
        return [s2[0]] + merge(s1, s2[1:])
```

1.12 Mario needs to jump over a sequence of Piranha plants, represented as a string of dashes (no plant) and P's (plant!). He only moves forward, and he can either step (move forward one place) or jump (move forward two places) from each position. How many different ways can Mario traverse a level without stepping or jumping into a Piranha plant? Assume that every level begins with a dash (where Mario starts) and ends with a dash (where Mario must end up):

Hint: You can get the ith character in a string `s` by using `s[i]`. For example,

```
>>> s = 'abcdefg'
>>> s[0]
'a'
>>> s[2]
'c'
```

You can find the total number of characters in a string with the built-in `len` function:

```
>>> s = 'abcdefg'
>>> len(s)
7
>>> len('')
0
```

```
def mario_number(level):
    """Return the number of ways that Mario can perform a sequence of steps
    or jumps to reach the end of the level without ever landing in a Piranha
    plant. Assume that every level begins and ends with a dash.

    >>> mario_number('-P-P-')   # jump, jump
    1
    >>> mario_number('-P-P--')   # jump, jump, step
    1
    >>> mario_number('--P-P-')  # step, jump, jump
    1
    >>> mario_number('---P-P-') # step, step, jump, jump or jump, jump, jump
    2
    >>> mario_number('-P-PP-')  # Mario cannot jump two plants
    0
    >>> mario_number('----')    # step, jump ; jump, step ; step, step, step
    3
    >>> mario_number('----P----')
    9
    >>> mario_number('---P----P-P---P--P-P----P-----P-')
    180
    """


        def ways(n):
        if n == len(level)-1:
            return 1
```

```
        if n >= len(level) or level[n] == 'P':
            return 0
        return ways(n+1) + ways(n+2)
    return ways(0)
```