CS 120: Intro to Computer Programming II

In-Class Activity - 05 Recursion - Day 2

Activity 1 - Turn in this one

Write a recursive function to count the number of "coins in a cup" - that is, the number of elements in an array. The `len` function is not allowed.

Solution:

```
def num_coins(data):
    if data == []:
        return 0
    return 1 + num_coins(data[1:])
```

Activity 2 - Turn in this one

Write a recursive function which returns the value of the **last** element in a linked list. (You may assume that the list is not empty.)

Solution:

```
def last_val(head):
    if head.next is None:
        return head.val
    return last_val(head.next)
```

Activity 3 - Turn in this one

Write a recursive function to count the number of **odd** values, in an array of integers.

Solution:

```
def count_odds(data):
    if len(data) == 0:
        return 0

    if data[0] % 2 == 0:
        return     count_odds(data[1:])
    else:
        return 1 + count_odds(data[1:])
```

Activity 4 - Turn in this one

OPTIONAL. Complete this if you have time, and turn it in. If you don't have time, you may report to your TA that you ran out of time.

Write a recursive function which returns `True` if a value k, passed as a second parameter, is found somewhere in a linked list. That is, the function will be of this form:

```
def search_linked_list(head, k):
```

Solution:

```
def search_ll(head, val):
    if head is None:
        return False
    if head.val == val:
        return True
    return search_ll(head.next, val)
```

Challenge Activity - Do not turn in this one

Write a recursive function that takes an array of values (not necessarily strings), and joins them all together into a single concatenated string, which it returns. For instance,

```
join_all([1,2,3,4,5])
```

should return `"12345"`, and

```
join_all(["aa","bb"])
```

should return `"aabb"`.

Solution:

```
def join(data):
    if len(data) == 0:
        return ""
    return str(data[0]) + join(data[1:])
```

Challenge Activity - Do not turn in this one

Write a recursive function that takes an array of values (of any type), and returns an array containing the same values, but reversed. That is,

```
recursive_rev([1,2,3])
```

should return `[3,2,1]`.

Solution:

```
def rev(data):
    if len(data) == 0:
        return []
    else:
        return rev(data[1:]) + [data[0]]
```