15-110 Hw3 - Written Portion

Name:

AndrewID:

#1 - Counting Down - 10pts

You want to write a countdown that runs fast, so you decide to count by 2's instead of 1's. For which values of n will this function run faster, and for which values will it not? In twenty words or less, why does this happen?

```
def countdown(n):
    if n == 0:
        print("Blastoff!")
    else:
        print(n)
        countdown(n - 2)
```

Answer:

#2 - 2D Lists - 10pts

Fill in the following table with the values in the 2D list returned by the mysteryFunction. Write an X in the squares that are outside the bounds of the list.

```
def mysteryFunction(x, y):
    myList = []
    for i in range(x):
        lst = []
        j = 0
        while j < y:
            if j <= i:
                lst.append("o")
            else:
                lst.append("-")
            j = j + 1
        myList.append(lst)
    return myList
matrix = mysteryFunction(6, 5)
```


#3 - Aliasing - 10pts

In twenty words or less, describe what the function below does. You may assume the inputs L1 and L2 are both lists.

```
def mystery2(L1, L2):
    temp = L1[0]
    L1[0] = L2[0]
    L2[0] = temp
    temp = L1
    L1 = L2
    L2 = temp
    return L1 + L2
```

Answer:

#4 - Aliasing and Mutability - 10pts

The following code creates and modifies lists. Determine each list's values after the code has run, then list any other variables that have the same id.

```
A = ["apple", "banana", "cantelope", "donut"]
B = A
B.remove("apple")
C = A + ["apple"]
C.insert(0, "eclair")
D = sorted(C)
```

Variable	List Values	Variables with Same id
A		
B		
C		
D		

Programming Problems

Each of these problems should be solved in the starter file available on the course website. They should be submitted to the Gradescope assignment Hw3 Full (Programming) to be autograded.

Most programming problems may also be checked by running the starter file, which calls the function testAll() to run test cases.

#1 - countFirstLetter(L, letter) - 5pts

Write a function countFirstLetter(L, letter) that counts and returns the number of elements whose first letter is the one given. You may assume the list elements are strings.
e.g., countFirstLetter(["dog", "daisy", "cat", "dandelion"], "d") returns 3

#2 - middleAppend(L, item) - 5pts

Write a function middleAppend(L, item) which finds the middle of the list (use integer division), inserts a new value at that location in the list, and returns the updated list.
e.g., middleAppend([1, 2, 3], 4) modifies the list to be [1, 4, 2, 3]

#3 - odds(L) - 5pts

Write a function odds(L) that returns a new list of only the odd-indexed items. Note that this is not the odd numbers- it is the odd **indexes**!
e.g., [0, 1, 2, 3, 4, 5] returns [1, 3, 5], and [1, 2, 3, 4, 5, 6] returns [2, 4, 6]

#4 - removeEvens(L) - 10pts

Write a function removeEvens(L) that removes the even-indexed items of the list, leaving a list of only the original odd-indexed items remaining in L.
e.g., [0,1,2,3,4,5] modifies the list to be [1,3,5] while [1,2,3,4,5,6] becomes [2,4,6]

Note: this is tricky because L is changing; you should check for aliasing issues. You can assume there is only one of each value in the list.

#5 - hiddenMessage(S) - 5pts

Write a function hiddenMessage(S) which takes a string, splits it by spaces, and returns a new string that is composed of the n-th letters of the n-th words.
e.g., "I'm here" returns "Ie", and "Come to office hours" returns "Cofr"

#6 - onlyPositive(L) - 10pts

Write a function onlyPositive(L) that inputs a list and returns a new list that contains only the positive elements of the original in the same order the original numbers occured. You may assume the list has only numbers in it.
e.g., [1, 2, 3] returns [1, 2, 3], [0, 1, 2] returns [1, 2], [-2, -1, 0] returns [ ]

#7 - reverseRecursive(L) - 10pts

Write a function reverseRecursive(L) that inputs a list and recursively returns a new list which has the elements in the original in reverse.
e.g., [1, 2, 3] becomes [3, 2, 1]

Note: you must use recursion to get full credit!s

#8 - maxRecursive(L) - 10pts

Write a function maxRecursive(L) that inputs a list and recursively returns the maximum value in the list. You may assume that len(L) >= 1.
e.g., maxRecursive([1, 2, 3]) returns 3, maxRecursive([2, 4, 6, 9, 10, 2, 6]) returns 10

Note: you must use recursion to get full credit!