Final Exam In-class Practice (focusing on harder string problems, lists, and 2d lists)

Reading code *(These problems are intended to be done on paper or in your head. Do not use a computer.)*

1. Suppose we have this function:

```
def weird(string):
    total = 0
    for pos in range(0, len(string)):
        digit = int(string[pos])
        total *= 10
        total += digit
    return total
```

What does this function return for `weird("12")`? For `weird("742")`? What does this function do in general?

2. Suppose we have this function:

```
def strange(str1, str2):       # Assume str1 and str2 are the same length.
    newstring = ""
    for pos in range(len(str1)):
        if str1[pos] < str2[pos]:
            newstring += str1[pos]
        else:
            newstring += str2[pos]
    return newstring
```

What does this function return for `strange("abcd", "dcba")`? What does this function do in general?

3. Suppose we have this function:

```
def funky(lst):
    for pos in range(0, len(lst)-1):
        if lst[pos] < lst[pos+1]:
            lst[pos] = lst[pos+1]
    return lst
```

What do `lst1` and `lst2` look like after this code runs?

```
lst1 = [1, 5, 2, 6, 3, 7]
lst2 = [1, 2, 3, 4, 5]
lst1 = funky(lst1)
lst2 = funky(lst2)
```

4. Suppose we have this function:

```
def silly(matrix):
    totalA = 0
    totalB = 0
    for row in range(0, len(matrix)):
        for col in range(0, len(matrix[0])):
            if matrix[row][col] > matrix[0][col]:
                totalA += 1
            if matrix[row][col] > matrix[row][0]:
                totalB += 1
    print(totalA, totalB)
```

Suppose we define a variable `m` like this:

```
m = [[1, 11, 8], [9, 5, 3], [4, 2, 12], [10, 7, 6]]
```

What does the function call `silly(m)` print? What does this function do in general?
Would the function behave any differently if we change the second `if` statement to `elif`?

Writing code *(This may be done on paper or on the computer. Go back and try the challenges only after you've solved all the regular problems.)*

1. Write a function called `ispal(string)` that returns `True` if `string` is a palindrome (reads the same forwards and backwards). The function returns `False` otherwise.

   ```
   def ispal(string):
   ```

   Examples:

   ```
   ispal("abccba") returns True     ispal("abcba") returns True
   ispal("a") returns True          ispal("ab") returns False
   ```

   CHALLENGE: Write a function called `iscopy(string)` that detects if a string consists of two copies of the same string back to back. e.g., `iscopy("yesyes")` returns `True`. `iscopy("aa")` returns `True`. `iscopy("bob")` returns `False`.

2. Write a function called `allsame(lst)` that takes a list argument and returns `True` if all the items in the list are the same, and False otherwise. You may assume `allsame` will never be called on an empty list.

   ```
   def allsame(lst):
   ```

   Examples:

   ```
   allsame([5, 5, 5]) returns True        allsame([5, 1, 5, 5]) returns False
   ```

3. Write a function called `slidedown(matrix)` that takes a 2d list argument. This function should change the matrix argument so that all the items in row 0 are moved into row 1, all the items in row 1 are moved into row 2, and so on. After this function is run, the items on the bottom row of the matrix will disappear entirely, and the items on the top row will appear twice (in row 0 and row 1).

   Example:

   ```
   m = [[1, 11,  8],
        [9,  5,  3],
        [4,  2, 12],
        [10, 7,  6]]

   m = slidedown(m)

   # m is now:  [[1, 11,  8],
                 [1, 11,  8],
                 [9,  5,  3],
                 [4,  2, 12]]
   ```

   CHALLENGE: Make 3 more functions that slide all the elements in a 2d list to the left, right, and upper directions.