Final Day

***if you skipped this class, you missed Upsorn buying us all candy
***losing points for misnaming the file on checkpoints, but very few so don't worry!
***make sure you understand everything in the old exam, email me (vlb9ae) if you can't figure one out
***when taking the exam, Manage your time !!
***don't write paragraphs, bullets are fine, this is not an English class so not grading on English
***if the question uses the word "return", it's a function, not a program

Practice of the Day 24:
http://www.cs.virginia.edu/~up3f/cs1110/practice-of-the-day/practice_24.txt

```
# trace through the code by hand
def func1(x, y):
    x = y
    return x


def func2(a, b):
    a[1] = 9
    try:
        print(a[func1(a[1], b)])
        x = [6, 6]
        a = [b, 7]
        return a

    except IndexError:
        print("IndexError")
    except ZeroDivisionError:
        print("ZeroDivisionError")
    except:
        print("any other errors")
    finally:
        print("finally")

x = [3, 4, 5]
b = 5
print(func2(x, b))
# func2 a = [3, 4, 5]; b = 5
# global a[1] = 9; a = [9, 4, 5] since it's a list (complex type)
# try to print(a[func1(a[1], b)])
    # func1(9, 5)
    # x = 5
    # return 5
# try print(a[5])
# there is no a[5], is "IndexError"
# print "Index Error
# print "finally"
# didn't return anything, so print None
```

Pass by reference

```
# final print:
# IndexError
# finally
# None

# if you run the code, this checks out
```

As always, if this doesn't make sense, Visualize Python will walk through it step by step

Try potd 25 and 26 at home using these principles

Practice of the Day 27:
http://www.cs.virginia.edu/~up3f/cs1110/practice-of-the-day/practice_27.txt

```
# Write code that takes a two dimensional list of integers
# of at least size two, and finds the minimum and
# next minimum integer in this two dimensional list.
# You are guaranteed that all numbers in the list are unique.
# Return the two numbers with a dash between them;
# for example, if the incoming list is [[2,3],[1,4]]
# you would return 1-2.
# You may not use any built-in functions/methods besides len()

# my solution
def mymins(list_of_lists_of_two):
    onelst = []
    for list_of_two in list_of_lists_of_two:
        for num in list_of_two:
            onelst.append(num)
    # easier to work with in a single list
    min = onelst[0]
    # initiate min
    for num in onelst:
        if num < min:
            min = num
            # update min if we find a smaller one
    onelst.remove(min)
    # take min out before looking for the next smallest
    min2 = onelst[0]
    # initiate next smallest
    for num in onelst:
        if num < min2:
            min2 = num
            # update next smallest if we find a smaller one
    # and return them both in the requested syntax
    return(str(min) + '-' + str(min2))

# class solution
def classmins(l1):
    min = l1[0][0]
```

```
    # initiate min to the first number
    for i in range(len(l1)):
        for j in l1(i):
            # iterating over every number
            if j < min:
                min = j
                # update min if we find a smaller one
    # practice by adding the next min to this code

# in the end, these four statement should all print the same thing
mymins([[1, 4], [2, 3]])
classmins([[1, 4], [2, 3]])
mymins([[2, 3], [1, 4]])
classmins([[2, 3], [1, 4]])
```