CSC 120 **ICA-28**

Work with your neighbor. (This will be graded for participation only.)

1. [White box testing] Write four unit tests for the function below:

```
# average(lst) -- returns the
# average of the numbers in lst.

def average(lst):
    sum = 0
    for i in range(len(lst)):
        sum += lst[i]

    return sum/len(lst)
```

ANS:

1) value for lst: `[]` → `errors out`
   tests 0 iterations of the loop and uncovers the divide-by-zero error

2) value for lst: `[6]` → `returns 6.0`
   tests 1 iteration of the loop

3) value for lst: `[2, 9, 6, 23, 0]` → `returns 8.0`
   tests > 1 iterations of the loop

4) value for lst: `[2, -4, -8, 0]` → `returns 2.5`
   tests > 1 iterations of the loop using neg values

2. [White box testing] Write four unit tests for the function below:

```
# Returns a list consisting of the strings in wordlist
# that end with tail.
def words_ending_with(wordlist, tail):
    outlist = []
    for item in wordlist:
        if item.endswith(tail):
            outlist.append(item)
    return outlist
```

ANS:

1) values for wordlist and tail:
   `['hi'], ''` → `returns ['hi']`
   Tests 1 iteration of the loop

2) values for wordlist and tail:
   `['bye', 'go', 'hope'],  'e'` → `returns ['bye','hope']`
   Tests > 1 iterations of the loop

3) values for wordlist and tail:
   `[], 'hi'` → `returns []`
   Tests 0 iterations of the loop

4) values for wordlist and tail:
   `'abcdef', 'c'` → `returns ['c']`
   Tests > 1 iterations of the loop

   Notice here that we passed in a string for the first argument rather than a list. The function works, but that may not have been the expectation.

3. [White box testing] The following code takes a list of numbers and compares pairs of consecutive numbers. For the pairs of numbers where the first is smaller than the second, it prints the corresponding indices.

   The function also prints those results based on the size of the list. If the list is large, it shows the results of the comparisons of a few initial consecutive numbers. Otherwise, it shows the results of all comparisons.

```
def my_function(lst):
   if len(lst)>5:
        for i in range(5):
            if lst[i]<lst[i+1]:
                 print("{}-th element is smaller \
                        than {}-th element".format(i, i+1))
    else:
        for i in range(len(lst)):
            if lst[i]<lst[i+1]:
                 print("{}-th element is smaller \
                        than {}-th element".format(i, i+1))
```

   a. Provide some test cases that will execute the if block.

ANS: [1, 2, 3, 4, 5, 6], [12, 23, 21, 54, 3, 56]

   b. Provide some test cases that execute the else block.

ANS: [], [1], [1, 5, 7]

   c. Now execute the test cases of the if block and check whether there are any errors.
ANS: The test cases do not create any errors.

d. Also, execute the test cases of the else block and check whether there are any errors.

ANS: the test cases [1] and [1, 5, 7] create errors.

e. Fix the errors if you find any.

ANS:

```
def my_function(lst):
   if len(lst)>5:
        for i in range(5):
            if lst[i]<lst[i+1]:
                 print("{}-th element is smaller
                        than {}-th element".format(i, i+1))
    else:
        for i in range(len(lst)-1): #fixes the error
            if lst[i]<lst[i+1]:
                 print("{}-th element is smaller
                        than {}-th element".format(i, i+1))
```

4. The following code moves the maximum element of the list `lst` to the end of the list:

```
def move_max_to_end(lst):
    for i in range(len(lst)-1):
        if lst[i]>lst[i+1]:
           t = lst[i]
           lst[i] = lst[i+1]
           lst[i+1] = t
```

Verify that the function is working properly by using an assert statement. Add the assert statement as the last line of the function. This assert will act as a postcondition to verify that the function behaves properly. You can define your own function for your assertion.

ANS:

```
def move_max_to_end(lst):
    for i in range(len(lst)-1):
        if lst[i]>lst[i+1]:
            t = lst[i]
            lst[i] = lst[i+1]
            lst[i+1] = t
    assert assert_function(lst), "move_max_to_end is not
                                    working properly"

def assert_function(lst):
    for i in range(len(lst)-1):
        if lst[i]>lst[len(lst)-1]:
            return False
    return True
```