CSC 185

HW 4 (36 pts)
Reading: Time Complexity: Chapter 2.
Optional reading: Review of Lists and Dictionaries: Chapter 1.8.

1. The function below was intended to swap the items in positions `i, j` of a list `myList`.

```
def swap(i, j, myList):
    temp = myList[i]
    myList.insert(i, myList[j])
    myList.insert(j, temp)
```

For example if `myList` is `["a", "b", "c", "d", "e", "f"]`, then `swap(1, 4, myList)` *should* change `myList` into `["a", "e", "c", "d", "b", "f"]`. However does not work as intended.

(a) (2 pts) **Debug:** Show the contents `myList` after each line for `swap(1, 4, ["a", "b", "c", "d", "e", "f"])` to see what goes wrong.
(b) (3 pts) **Design:** Rewrite the `swap` method so that is behaves correctly and does so without creating a new list. I highly recommend writing the code on paper (as a way to practice for Exams) and then checking your code with IDLE.
(c) (2 pts) Report the time complexity of your `swap` function in Big-O notation in terms of *n*, the size of `myList`.

2. (a) (3 pts) Write a function `maxValue` that takes two `lists A` and `B`, and returns the maximum number among both. **Your code must use loops.** For example:

```
maxValue([0, 5, 2], [6, 4, 1]) should return 6
maxValue([1, 1, 1], [1, 0, 0, 4, 9, 110, 23]) should return  110
```

I highly recommend writing the code on paper (as a way to practice for Exams) and then checking your code in IDLE.
(b) (2 pts) Report the time complexity of your `maxValue` code in Big-O notation in terms **both** *n*, the size of `A`, **and** `m` the size of `B` as those are the inputs to your method.

3. Consider the following code that is intended to remove all items in a `list`, named `myList`.

```
def func(myList):
  while i < len(myList):
    myList.pop(i)
    i=i+1
```

(a) (2 pts) Assume `myList` contains: `[ "toad", "coat","worm", "maple", "coat", "garage"]`. What are the contents of the list after **each** iteration of the loop?
(b) (2 pts) Report the time complexity of the function in Big-O notation, using `n` to denote of the size of `myList`.

4. Assume `word` is a string, and `x` is a character.

```
def myfunc(word, x):
  mydictionary={}
  for w in word:
    if w in mydictionary:
      mydictionary[w] = mydictionary[w]+1
    else:
      mydictionary[w]=1
  return mydictionary[x]
```

(a) (1 pt) What will be returned by `myfunc("HELLO LEMOYNE", 'L')`?
(b) (2 pts) In one sentence describe what the function is computing in relation to `word` and `x`
(c) (3 pts) State the Big-O running time of the function in terms of `n`, the length of word.

5. (10 pts) State the Big-O running time of the following code snippets in terms of *n*?

(a)
```
x = 0
for i in range(n):
     x = x + i
for i in range(n):
     x = x + i
```

(b)
```
x = 0
for i in range(100*n):
     x = x + i
```

(c)
```
x = 0
for i in range(n):
     for j in range(200):
          x = x + i + j
```

(d)
```
i = 0
while i<n:
  i = i+2
```

(e) Hint: Check how many times the loop executes for some values of n such as n=8, n=128, n=1024.

```
i=n
while i>1:
     i = i/2
```

6. (4 pts) For each function below let `n` be the size of the data structure given to the function. State the running time of each function in Big O notation in terms of `n`. Remember to consider the running time of the methods being called.

(a) Assume `n` is an int

```
def populateList(n):
   alist = []
   for i in range(n):
      alist.append(i)
   return alist
```

(b) Assume `n` is an int

```
def populateList(n):
   alist = []
   for i in range(n):
      alist.insert(i,i)
   return alist
```

(c) Assume `myList` is a list and `i,j` are integers

```
def getLargerValue(mylist, i, j):
      ans = 0
      if mylist[i] > mylist[j]:
            ans = mylist[i]
      else:
            ans = mylist[j]
      return ans
```

(d) Assume `d` is a dictionary that maps strings to numbers

```
def getLargerValue(d, i, j):
      ans = 0
      if d[i] > d[j]:
            ans = d[i]
      else:
            ans = d[j]
      return ans
```