fullName:_____andrewID:_____
recitationLetter:_____ Lecture (1, 2, or 3)_____

# Quiz9 version A

You **MUST** stop writing and hand in this **entire** quiz when instructed in lecture.

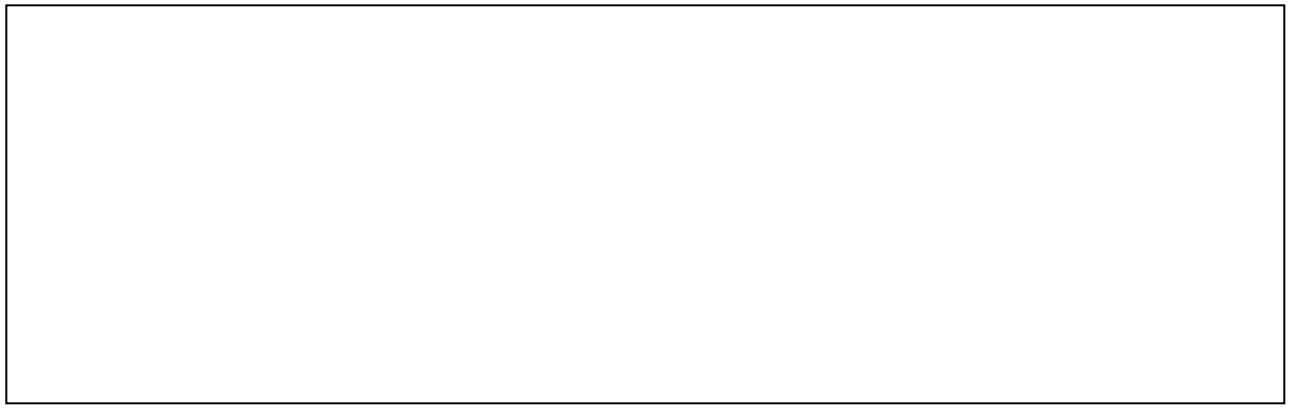
- You may not unstaple any pages.
- You may not use your own scrap paper. If you must use additional scrap paper, raise your hand and we will provide some. You must hand this in with your paper quiz, and we will not grade it.
- Failure to hand in an intact quiz will be considered cheating. Discussing the quiz with anyone in any way, even briefly, is cheating.
- Write "q" next to this line for one bonus point.
- You may not use any concepts we have not covered in the notes this semester. We may test your code using additional test cases. Do not hardcode. Assume almostEqual(x, y) and roundHalfUp(n) are both supplied for you. You must write all other helper functions you wish to use.

## CT1: Code Tracing [15pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct1(s):
    if len(s) <= 1:
        return s
    else:
        i = len(s)//2
        return s[i] + ct1(s[i+1:]) + ct1(s[:i])

print(ct1('35126'))
```



## CT2: Code Tracing [15pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct2(n):
    if n < 10:
        return [n]
    else:
        return [n%10] + ct2(n//100)

print(ct2(12345))
```

## Free Response 1: secondLargest(L) [35pts]

Note: for all FR's on this quiz, you must not use for or while loops. Solutions that use loops will earn zero points.
You also must not use any builtin functions or methods except those that run in O(1) time with respect to the length of the problem's input. Penalties for slower builtins (such as min(L) and max(L)) range from -50% to zero credit. List slicing is allowed. Also, checking the length of a list or string is O(1).

Write the recursive function secondLargest(L) that takes a list of integers and returns the second-largest integer in the list, without sorting the list. Thus:

```
assert(secondLargest([2,5,3,4,1]) == 4)
```

If the list has duplicate values, it is possible for the largest and second-largest values to be the same. Thus:

```
assert(secondLargest([6,5,6,6,6]) == 6)
```

If the list does not have a second-largest value, return None. Thus:

```
assert(secondLargest([1]) == None)
assert(secondLargest([]) == None)
```

Hint: you may find it helpful to use a wrapper function.

## Free Response 2: longestVowelRun(s) [35pts]

Note: for all FR's on this quiz, you must not use for or while loops. Solutions that use loops will earn zero points.
You also must not use any builtin functions or methods except those that run in O(1) time with respect to the length of the problem's input. Penalties for slower builtins (like calling s.lower() or s.isupper()) range from -50% to zero credit. String slicing is allowed. Also, checking the length of a list or string is O(1).

Write the recursive function longestVowelRun(s) that takes a string s and returns the length of the longest run of consecutive vowels in s, ignoring case, where the vowels are A, E, I, O, and U.
For example:

```
assert(longestVowelRun('AbcAeiD aEoa ba') == 4) # aEoa
assert(longestVowelRun('cnsnts') == 0) # no vowels
assert(longestVowelRun('aeiouAEIOU') == 10) # all vowels
```

Hint: we found it very helpful to write the helper function vowelRun(s) that returns the vowel run for the string s starting only from the beginning of s. For example, vowelRun('abaeiou') returns 1, and vowelRun('daeiou') returns 0, and vowelRun('aeiobu') returns 4.

## bonusCT: Code Tracing [2pts]

This question is optional. Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def bonusCt(L):
    def f(L):
        return L if L == [ ] else [L[0]//2]*2 + f(L[1:])
    if (set(L) == {0}):
        return len(L)
    else:
        return bonusCt(f(L))
print(bonusCt([2,8]))
```