Name:_____ Section:___ Andrew Id: _____

15-110 Spring 2019 Quiz5

*** 30 minutes, No calculators, no notes, no books, no computers, no phones**

1. **Code Tracing [5 pts]** Indicate what the following program prints. Place your answer in the box.

```
def ct1(L):
    s = set()
    for i in range(len(L)):
        s.add(i + L[i])
    return s - set(L)
print(ct1([ 4, 3, 7, 4]))
```

2. **Code Tracing [5 pts]** Indicate what the following program prints. Place your answer in the box.

```
def ct2(d):
    s = set()
    for key in d:
        if (d[key] in d):
            s.add(d[key])
    return s
print(ct2({1:1, 2:4, 3:6, 4:6, 5:4}))
```

3. **Short Answers [15 pts; 3 pts each value]**

1. Which of the following describes how most computers (including the LMC computer) operate at the lowest level?

A) The read-eval-print loop

B) The fetch-decode-execute cycle

C) The see-say-do pattern

D) The load-analyze-display program

2. Which of these languages is at the lowest level?

A) Python

B) Machine Language

C) Assembly Language

D) C

3. If there is no branching, what happens to the Program Counter after one instruction is executed?

A) Nothing. It stays the same.

B) There is no single answer -- it depends on the instruction.

C) It increases by 1.

D) It resets to 0.

4. Registers are a kind of memory. What distinguishes registers from the main memory in RAM?

A) Registers are on the CPU.

B) Registers are faster to use.

C) There are far fewer registers than memory locations in RAM.

D) All of the above.

5. What will the following LMC Assembly Language program output when it is run with an input of 5?

```
INP
STA 99
ADD 99
STA 99
ADD 99
OUT
HLT
```

4. **Free Response: TalkativeCow class [35 pts]**

Write the class TalkativeCow so that the following test code passes (without hardcoding, so any reasonable edits to the test code still pass):

```
def testTalkativeCowClass():
    print('Testing TalkativeCow class...', end='')
    f1 = TalkativeCow(2) # start with 2 moo's!' (You can assume a positive number)
    assert(f1.moo() == 'Moo!Moo!') # note: no spaces!
    assert(f1.moo() == 'Moo!Moo!Moo!') # one more moo each time!
    assert(f1.moo() == 'Moo!Moo!Moo!Moo!') # one more moo each time!
    f2 = TalkativeCow(1) # start with 1 moo!'
    assert(f2.moo() == 'Moo!')
    assert(f2.moo() == 'Moo!Moo!')
    print('Passed!')
```

.

5. **Free Response: closeButNotQuite(L) [40 pts]**

Write the function closeButNotQuite(L) that takes a list L of integers (of any size, maybe very large and maybe very small), and returns a set of all the integers that are within 1 (inclusive) of some value in L, but which themselves are not in L.

For example:

```
L = [ 1, 2, 3]
assert(closeButNotQuite(L) == { 0, 4 })
```

And:

```
L = [ -1234, 5678 ]
assert(closeButNotQuite(L) == { -1235, -1233, 5677, 5679 })
```

For up to 2 points of bonus, write the function so that it runs in O(N) where N == len(L).

6. **Bonus Code Tracing [2 pts]** Indicate what the following program prints. Place your answer in the box.

```
def bonusCt1(s): return len(set(str(s)))
print(bonusCt1(set('abacab' * len('abacab'))))
```

7. **Bonus Code Tracing [2 pts]** Indicate what the following program prints. Place your answer in the box.

```
def bonusCt2(n):
  def f(n): return n if (n < 2) else f(n-2) + 4*(n-1)
  return len(str(f(n)))
print(bonusCt2(1010))
```