fullName: _____ andrewID: _____ section: ____

15-112 F26 Quiz4 Version A
Time: 30 Minutes

You must write your name on this paper and hand this back in immediately after the assessment. If we do not receive it immediately, you will receive a zero on the assessment. Do not unstaple any pages. All pages must be handed in intact.

Do not use your own scrap paper. You should not need it, but if you must absolutely have scrap paper, raise your hand and we will provide some. Write your andrewID clearly on it and hand it in with your quiz. We will not grade anything on scrap paper.

You may not ask questions during the quiz, except for English-language clarification questions. If you are unsure about a problem, take your best guess.

Before and during the quiz, you may not view any other notes, prior work, websites or resources, including any form of AI. You may not use calculators, phones, laptops, or any other devices. You may not communicate with anyone else except for current 112 TAs or faculty during the assessment. All syllabus policies apply.

You may not discuss this test with anyone else, even briefly, in any form, until we have released grades. Failure to abide by these rules may result in an academic integrity violation.

Do not use material we have not yet covered. Thus, do not use sets, dictionaries, OOP, or recursion. Also: do not use eval() or exec().

For all problems, assume that the required imports (such as 'import math') are present.

Do not open this or look inside (even briefly) before you are ready to begin. Do not spend more than the specified time noted above on this assessment.

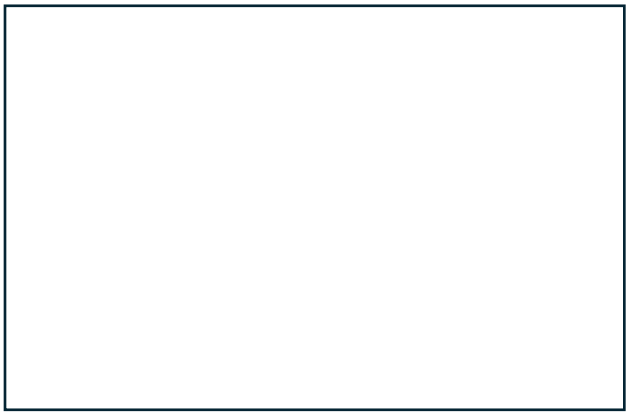
Code Tracing [10 pts]

Indicate what the following code prints. Place your answer (and nothing else) in the box below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

```
import copy

def ct1(L, d):
    M = L
    N = copy.copy(L)
    M.append(d)
    L = [d] + L
    M = M[:-1]
    L.append(N.pop())
    print(f'L={L}')
    print(f'M={M}')
    print(f'N={N}')

L = [3]
ct1(L, 4)
print(f'L={L}') # don't miss this
```



Fill in the Blank [15 pts, 7.5 pts each]

FitB1: [7.5 pts]
Fill in the blank so f(L) == g(L) for all lists L where the values in L are all integers.

```
def f(L):
    assert(len(L) > 0)
    x = y = L[0]
    for v in L:
        if v > x:
            x = v
        elif v < y:
            y = v
    return x - y

def g(L):
    assert(len(L) > 0)

    return _____________________________________________________________
```

FitB2: [7.5 pts]
Fill in the blank so f(L) == g(L) for all lists L where the values in L are all integers.

```
def f(L):
    result = 0
    for v in L:
        x = v ** 2
        result += x
    return result

def g(L):
    return _____________________________________________________________
```

FR1: mutatingRemoveStrings [15 pts]

Write the function mutatingRemoveStrings(L) that takes a possibly-empty list L of arbitrary values, mutatingly removes all the strings in L, and returns a list of all the removed strings in their original order.

For example:

```
L = [1, 2, 'yes', 3, 'no']
assert(mutatingRemoveStrings(L) == ['yes', 'no'])
assert(L == [1, 2, 3])
```

And this example where L does not contain any strings:

```
L = [1, 2]
assert(mutatingRemoveStrings(L) == [ ])
assert(L == [1, 2])
```

Write your solution on the next page.

FR2: runCode [60 pts]

Background: for this problem, you will run code, similar to runSimpleProgram, but much simpler. Here, variable names must be single lowercase letters. At the start of the program, every variable is set to 0.

Ignoring leading whitespace, every line of code is one of these:

- A comment, which starts with !, and is ignored,
- A blank line, which is ignored, or
- A line of the form:
  `variable = variable + value`
  where the value is a non-negative int.

Here is an example of legal code in this simple language:

```
code = """
          y = y + 32
          ! Now: y = 32

          x = y + 5
          ! Now: x = 37, y = 32

          x = x + 100
          ! Now: x = 137, y = 32
       """
```

With that, write the function runCode(code) that takes a multiline string of legal code, runs the code, and returns a list of all the non-zero variables (as tuples of the form (name, value)) in alphabetical order.

For example, from the code above:

```
assert(runCode(code) == [('x', 137), ('y', 32)])
```

Hint: since variables must be single lowercase letters, there are at most 26 unique variables in this language. And again, all of them start set to 0.

Write your solution on the next page.

You may continue your solution on the next page if needed.

BonusCT

These CTs are optional, and intended to be very challenging. They are worth very few points. Indicate what the following code prints. Place your answers (and nothing else) in the boxes below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

bonusCT1: [1pt]

```
def bonusCt1(L):
    def f(L):
        s = str(L)
        while ', ' in s:
            s = s.replace(', ', '')
        return int(s[1:-1])
    while f(L) < 12345:
        L.append(f(L))
    return L
print(bonusCt1([2,3]))
```

bonusCT2: [1pt]

```
def bonusCt2(n, d):
    L = [ 3 ]
    while len(L) < n:
        L.append((sum(L) + 1) % 10)
    while len(L) > 1:
        i = j = 0
        for k in range(len(L)):
            if L[k] < L[i]: i = k
            if L[k] > L[j]: j = k
        for k in sorted([i, j], reverse=True):
            L.pop(k)
    return L[0] * n + d
print(bonusCt2(5, 7))
```

This page is intentionally blank