fullName: _____ andrewID: _____ section: ____

15-112 F25 Quiz3 Version 1B
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 sets, dictionaries, or recursion.

Do not open this or look inside (even briefly) before you are ready to begin. Do not spend more than 30 minutes on this assessment.

Code Tracing

CT1: [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, n):
    v = L[1] * n
    L = L + [v] * min(L)
    L.insert(1, f'y{L.pop()*2}')
    L.remove(20)
    return L
L = [4, 5, 3]
print(ct1(L, 4))
print(L == [4, 5, 3]) # do not miss this
```

CT2: [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.

```
def ct2(s):
    L = list(s)
    M = [ ]
    for i in range(1, len(L)):
        N = L[i:-i:i]
        s = ''.join(N)
        M.append(s)
        if s == '':
            break
    return M
print(ct2('bcdefgh'))
```

CT3: [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 ct3(L):
    M = L
    N = copy.copy(L)
    L *= 2
    M = M + [7]
    L += [8]
    L = sorted(L, reverse=True)
    N.append(9)
    print(L)
    print(M)
    print(N)
L = [6]
ct3(L)
print(L) # don't miss this
```

Fill in the Blank (FitB)

FitB1: mutatingSortedEvensAndOdds [10 pts]

You are given the function mutatingSortedEvensAndOdds(L, M) that solves the following problem, only with two parts removed. You need to fill in the blanks with the missing code.

Note: when you fill in blanks, you must not add any newlines and you must not use any semicolons..

With that, here is the writeup for mutatingSortedEvensAndOdds:

The function mutatingSortedEvensAndOdds(L) takes a possibly-empty list L of integers and mutatingly changes L so that after the call the even values in L all occur first, in sorted order, followed by the odd values in L, also in sorted order. The function should return None.

For example:

```
    L = [ 3, 4, 6, 1, 5, 2 ]
    assert(mutatingSortedEvensAndOdds(L) == None)
    assert(L == [2, 4, 6, 1, 3, 5])
```

And here is the FitB solution (fill in the two missing blanks):

```
def mutatingSortedEvensAndOdds(L):
    # first mutatingly remove odds
    odds = [ ]
    i = 0
    while i < len(L):
        if L[i] % 2 == 1:
            odds.append(_____________________________________) # <- BLANK #1
        else:
            i += 1
    # now sort the evens
    L.sort()
    # now mutatingly add the sorted odds

    ____________________________________________________  # <- BLANK #2
```

FR: averageScores [60 pts]

Background: A "grade report" is a 1d list L like so:

```
['Ann', 92, 88, 'Bob', 80, 84, 88, 'Cal', 42, 'END']
```

The list contains a name, followed by 1 or more scores for that person, followed by another name, and 1 or more scores, and so on, until the last value, which is always 'END'. You can assume that the list always contains at least one name, names are unique, every name contains at least one score, the scores are all non-negative integers, and 'END' is always the last value in the list.

With that, write the non-mutating function averageScores(L) that takes a grade report as just described and returns a new list with each student's name (in the order they appeared) followed by their average score (rounded to the nearest int). Do not include 'END' in this result.

In the example above:

- Ann scored 92 and 88, which average to 90.
- Bob scored 80, 84, and 88, which average to 84.
- Cal scored 42, which averages to 42.

Thus:

```
L = ['Ann', 92, 88, 'Bob', 80, 84, 88, 'Cal', 42, 'END']
assert(averageScores(L) == ['Ann', 90, 'Bob', 84, 'Cal', 42])
# And verify non-mutating:
assert(L == ['Ann', 92, 88, 'Bob', 80, 84, 88, 'Cal', 42, 'END'])
```

This example shows rounding:

```
L = ['Deb', 90, 91, 94, 'END']
# Note that (90+91+94)/3 == 91.666..., which we round to 92.
assert(averageScores(L) == ['Deb', 92])
```

Note: you can use round() or rounded() (or neither).

Write your solution on the following pages.

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):
        a = L[-1]
        if L == [a]:
            return (a,a)
        else:
            b,c = f(L[:-1])
            return (a,c) if a<b else (b,a) if c<a else (b,c)
    a,b = f(L)
    while a and b: a,b = a+1,b+1
    return b
print(bonusCt1([3, -4, 6, -8, 12, -14]))
```

bonusCT2: [1pt]

```
def bonusCt2(s):
    L = s.split()
    M = [L[1].count(L[2][1])] * 456
    for i in range(0, len(L), 2):
        for c in L[i]:
            M[ord(c)] += 1
    N = sorted([[str(M[i]), chr(i)] for i in range(len(M))
                if M[i]], reverse=True)
    return '-'.join([''.join(s) for s in N])
print(bonusCt2('ALL IS WELL HERE'))
```