fullName: _____ andrewID: _____ section: ____

15-112 F25 Quiz3 Version 2B
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: mutatingDeltas [10 pts]
You are given the function mutatingDeltas(L) 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 mutatingDeltas:

The function mutatingDeltas(L) takes a list L of integers, where L contains at least 2 non-zeros. It first mutatingly removes all the 0s in L, then mutatingly replaces the remaining values in L (except the first value, which is removed) with the delta (that is, the difference) from that value to the preceding value. For example:

```
L = [2, 3, 5, 4, 1]
assert(mutatingDeltas(L) == None)
assert(L == [1, 2, -1, -3])
```

Since 3 - 2 == 1, the first value in L is 1. Since 5 - 3 == 2, the second value in L is 2. Since 4 - 5 == -1, the third value in L is -1. Since 1 - 4 == -3, the fourth value in L is -3.

As another example:

```
L = [0, 2, 0, 0, 3, 5, 4, 0, 0, 0, 1, 0]
assert(mutatingDeltas(L) == None)
assert(L == [1, 2, -1, -3])
```

Since 0s are removed, this reduces to the same result as the previous example.
And here is the FitB solution (fill in the two missing blanks):

```
def mutatingDeltas(L):
    # Note: assume L contains at least 2 non-zeros
    # first mutatingly remove the 0's
    i = 0
    while i < len(L):
        if L[i] == 0:

            _____________________________________________  # <- BLANK #1
        else:
            i += 1
    # now replace values with deltas
    prev = L.pop(0)
    for i in range(len(L)):

        prev, L[i] = _____________________________________ # <- BLANK #2
```

FR: getTeamRecord [60 pts]

Background: for this problem, we will be given a list of results from baseball games (where every game always has a winner and a loser, no ties). For example, consider these results, where the home team is listed first in each score:

```
Cubs  0 - Bucs  2  # Bucs win, Cubs lose
Bucs 14 - Mets 15  # Mets win, Bucs lose
Mets  6 - Reds  1  # Mets win, Reds lose
Reds  3 - Bucs  4  # Bucs win, Reds lose
```

We will represent these results in a single list reading team names and scores in the order they appear in the results. For the results above, that would be:

```
[ 'Cubs',  0, 'Bucs',  2,
  'Bucs', 14, 'Mets', 15,
  'Mets',  6, 'Reds',  1,
  'Reds',  3, 'Bucs',  4 ]
```

Note that we added whitespace to make that clearer, but really it's just a normal 1d list.

With that in mind, write the function getTeamRecord(results, team) that takes a 1d list of results (as just described) and the name of one team, and returns the record of that team as a tuple of (wins, losses).

For example:

```
results = [ 'Cubs',  0, 'Bucs',  2,
            'Bucs', 14, 'Mets', 15,
            'Mets',  6, 'Reds',  1,
            'Reds',  3, 'Bucs',  4 ]
assert(getTeamRecord(results, 'Cubs') == (0, 1)) # 0 win, 1 loss
assert(getTeamRecord(results, 'Bucs') == (2, 1)) # 2 win, 1 loss
assert(getTeamRecord(results, 'Mets') == (2, 0)) # 2 win, 0 loss
assert(getTeamRecord(results, 'Reds') == (0, 2)) # 0 win, 2 loss
assert(getTeamRecord(results, 'Jays') == (0, 0)) # 0 win, 0 loss
```

You can assume that the results list is non-empty and in the correct format.

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'))
```