fullName:_____ andrewID:_____ section:___

15-112 N25
Quiz4 Part 2 of 2

This is the paper version of **Part 2 of Quiz4**. You will receive this paper version only if requested (or if taking the quiz with the testing center). You **must write your name on this paper and hand this back** in immediately after the assessment, even if you do not use it to write your answers. If we do not receive it immediately, you will receive a zero on the assessment.

Quiz4 Part 2 is located in section 7.9 of CS Academy (the end of Unit 7), and contains only the FRs. If you cannot see the submit button, call over a TA or Mike and let us know your andrewID.

We will not grade anything you write on these pages unless you initial the box below:

_____ **Write your initials here if and only if you wish for us to grade your FRs on this paper *instead of* any answers submitted in CS Academy.** Leave the space blank if you wish for us to grade what is written in CS Academy (the default, recommended option).

During the quiz, as always, you may not view any other notes, prior work, websites or resources, including any form of AI. You may not communicate with anyone else except for TAs or faculty during the assessment. All syllabus policies apply.

Some students will take this at a different time with testing accommodations. As such, 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 recursion on these FRs, or anything else disallowed prior to Unit 7 the recursion unit. (Note that you *will* need to use recursion later, when we do WS5)

Do not open this or look inside (even briefly) before we instruct you to begin. Close it once you are done.

Part 2[70pts total]: Free Response

Your functions should work generally for the kinds of inputs specified in the problem statement, and we may test your code using additional test cases. We will manually grade both of these problems for partial credit if you do not pass all the test cases.

FR1[35pts]: nonmutating insertRowCol(L, row, col, val)

Write the nonmutating function `insertRowAndCol(L, row, col, val)` which takes a rectangular 2d list L, and returns a new list with the same values of L, along with one new row and one new column at the locations specified by `row` and `col`. The cells in the new row and column are all set to `val`.

For example, if L is the following list:

```
[[1,  2,  3,  4],
 [5,  6,  7,  8],
 [9, 10, 11, 12]]
```

Then `insertRowAndCol(L, 1, 2, 42)` should return:

```
[[ 1,  2, 42,  3,  4],
 [42, 42, 42, 42, 42],
 [ 5,  6, 42,  7,  8],
 [ 9, 10, 42, 11, 12]]
```

In the example above, a row is inserted at index 1, and a column is inserted at index 2. All the values in the inserted row and column are set to 42.
You are guaranteed that `row` and `col` will be valid indexes for L.

Remember, you must not mutate L!

Here are some test cases:

```
    L = [[1,  2,  3,  4],
         [5,  6,  7,  8],
         [9, 10, 11, 12]]
    assert(insertRowAndCol(L, 1, 2, 42) == [[ 1,  2, 42,  3,  4],
                                            [42, 42, 42, 42, 42],
                                            [ 5,  6, 42,  7,  8],
                                            [ 9, 10, 42, 11, 12]])

    L = [[1, 2, 3],
         [4, 5, 6]]
    assert(insertRowAndCol(L, 0, 0, 5) == [[5, 5, 5, 5],
                                           [5, 1, 2, 3],
                                           [5, 4, 5, 6]])
```

More on the next page

```
# Continued from previous page
    L = [[1, 2, 3],
         [4, 5, 6]]
    assert(insertRowAndCol(L, 1, 3, 5) == [[1, 2, 3, 5],
                                           [5, 5, 5, 5],
                                           [4, 5, 6, 5]])

    # Verify that the function is non-mutating
    L = [[1,  2,  3,  4],
         [5,  6,  7,  8],
         [9, 10, 11, 12]]
    insertRowAndCol(L, 1, 2, 42)
    assert(L == [[1,  2,  3,  4],
                 [5,  6,  7,  8],
                 [9, 10, 11, 12]])

  #Begin your answer here or on the next page
```

```
# Begin or continue your answer here
```

FR2[35 pts]: getFlightTable(flightInfo)

Background: this problem starts with a multiline string of flight times between cities, like so:

```
flightInfo = '''
From Pittsburgh to LA takes 5.5 hours
From LA to Seattle takes 3 hours
From Seattle to Pittsburgh takes 4.5 hours
'''
```

Every non-empty line in the flightInfo will always be of the form:
`"From {city1} to {city2} takes {time} hours"`

We can convert this string into a 2d dictionary of flight times like so:

```
{'Pittsburgh': {'LA':5.5, 'Seattle':4.5},
 'LA': {'Pittsburgh':5.5, 'Seattle':3.0},
 'Seattle': {'LA':3.0, 'Pittsburgh':4.5} }
```

The outer dictionary contains departure cities as keys, and contains a dictionary as a value that contains key-value pairs of arrival cities, and flight times. Important notes:

- Flight times are also the same in reverse. If a string lists a time for a flight from city A to city B, the time is the same for a flight from city B to city A. Both the flight listed, and its reverse, should be added to the flight dictionary.
- The times in the inner dictionaries are always floats, even if they were ints in the string.
- You can assume that any from/to city pair occurs no more than once in the string.
- You can assume that the string contains at least one non-blank line.
- You can assume that city names do not contain any spaces.
- You should ignore blank lines in the string.

With this in mind, write the function **`getFlightTable(flightInfo)`** that takes a multiline string of flight times and returns a 2d dictionary of flight times, as just described.

Hint: you may want to use both `s.splitlines()` and `s.split()` here (both of which produce lists that can be looped through or indexed into).

See the additional test cases on the following page:

```
flightInfo = '''
From Pittsburgh to LA takes 5.5 hours
From LA to Seattle takes 3 hours
From Seattle to Pittsburgh takes 4.5 hours'''

assert(getFlightTable(flightInfo) ==
{'Pittsburgh': {'LA':5.5, 'Seattle':4.5},
 'LA': {'Pittsburgh':5.5, 'Seattle':3.0},
 'Seattle': {'LA':3.0, 'Pittsburgh':4.5}
})

flightInfo2 = '''
From NY to Boston takes 1.25 hours
From Scranton to NY takes 0.5 hours'''

assert(getFlightTable(flightInfo2) ==
{'NY': {'Boston':1.25, 'Scranton':0.5},
 'Boston': {'NY':1.25},
 'Scranton': {'NY':0.5}
})
assert(getFlightTable(flightInfo2)['NY']['Scranton'] == 0.5)

flightInfo3 = '''
From Shanghai to Shenzhen takes 2.5 hours
From Chongqing to Shanghai takes 2.25 hours
From Shenzhen to Chongqing takes 2 hours
From Chongqing to Qingdao takes 4.25 hours'''

assert(getFlightTable(flightInfo3) ==
{'Shanghai': {'Shenzhen':2.5, 'Chongqing':2.25},
 'Shenzhen': {'Shanghai':2.5, 'Chongqing':2},
 'Chongqing': {'Shanghai':2.25, 'Shenzhen':2, 'Qingdao':4.25},
 'Qingdao': {'Chongqing':4.25}
})
assert(getFlightTable(flightInfo3)['Shenzhen']['Shanghai'] == 2.5)
assert(getFlightTable(flightInfo3)['Shanghai']['Shenzhen'] == 2.5)
```

Write your answer on the following page

```
# Write your answer here
```

```
# You may continue your answer here if you wish
```