MSIS 640 Final Exam

There are two questions in this exam. Please use q1.py and q2.py as your file names for the three questions. Following are tips and rules for taking the exam:

- *Any cheat such as copying code from another person, will fail this course.*
- Design first, you should know how to do it in your mind and break the tasks into small steps.
- Work incrementally, code one step at a time, and test the step.
- Use VS Code as your editor. It helps you fix syntax errors.
- Run the code before submission. If the code doesn't run, i.e., has syntax error, the highest score is 20% of the total score for the question.
- You can use your book and Internet to search help and check syntax. Don't ask any person.
- You can use any Python library and package if it helps. However, all questions can be done without any 3rd party code.
- When you complete both questions, please submit both files together to Canvas. You can submit multiple times but please submit the latest version of q1.py and q2.py together. Old submission will be discarded. We can only grade your exam by running the source code files.
- Screenshot or another file format are not acceptable. Wrong files get 0.

Question 1 (100 points for q1.py)

A company has one top level department. Each department may have many subdepartments. Each department is represented as a list that has one manager and lists of subdepartments. In the lowest level department, there is only one manager and a list of staffs. Each level has a four-space indentation. Below are two examples.

Example 1, the following code represent a small company that has only one top level department. It has a manager John and two staffs Tom and Cindy. Staffs are stored in a list followed by its manager.

```
com_1 = [
    'John',
    ['Tom', 'Cindy']
]
```

You should print the department hierarchy as the following – staffs are indented 4 spaces:

```
John
    Tom
    Cindy
```

Example 2: The following code has one top-level department (the manager is Bill) and two sub-department. The first subdepartment (the manager is John) has two staffs (Tom and Cindy). The second department has one subdepartment (the manager is Cindy) that has only one subdepartment (the manager is Alice) that has two staffs (Bob and Tyler)

```
com_2 = [
    'Bill',
    ['John', ['Tom', 'Cindy']],
    ['Rambo', ['Cindy', ['Alice', ['Bob', 'Tyler']]]],
]
```

You should print the department hierarchy as the following – each subdepartment and staffs have a four-space indentation.

```
Bill
    John
        Tom
        Cindy
    Rambo
        Cindy
            Alice
                Bob
                Tyler
```

Please write a function print_tree(company) that print a given company hierarchy that may have arbitrary levels of subdepartment. You can use the following code as a starting template. We will test your code with different company hierarchy data.

```
com_1 = [
    'John',
    ['Tom', 'Cindy']
]

com_2 = [
    'Bill',
    ['John', ['Tom', 'Cindy']],
    ['Rambo', ['Cindy', ['Alice', ['Bob', 'Tyler']]]],
]

def print_tree(company):
    pass # implement your function here

print_tree(com_1)
print_tree(com_2)
```

Tips:

- Because there could be any levels of subdepartment, you may want to use a recursive function to print the company hierarchy.
- A department has only two types of elements, a str and one or more list. You can use isinstance(var, str) to check if a list element is a str or not.
- Because the department level is used in the result for indentation, you may want to define a function that takes two parameters: a list and its current level. The level increases in each recursion.

Question 2 (100 points for q2.py)

We have a text file (attached in this folder) scores.txt that has student test scores as the following:

name,test1,test2,test3,test4
John,80,70,85,92
Bill,75,63,92,76
Cindy,95,98,,92
Alice,87,73,29,98
Bob,92,null,missing,98
David,92,37,77,?

We will use a different test score file to check your program. The test file may have different number of tests and different values to represent missing score. Your program should be able to process the input file as the following:

1) The first line is student name and test names. A score file may have different number of tests (test5, test 6 etc.). Your program should be able to handle any number of tests. All fields are separated by a comma , separator.
2) A student may miss a test. However, the missing value could be empty (Cindy's test3), invalid number value (Bob's test2 of null, test3 of missing, David's test4 of ?). A simple rule is that if you can not convert the value into an integer, then it is a missing value. Missing values are ignored; therefore each student may have a different number of tests.
3) Your program calculates the number of tests and final grade for each student. For example, in the above test, Bob only two tests and has a final grade of A. The grading rules are
    a. Ignore the missing grades
    b. Calculate the average score
    c. Assign a final grade according to the following rule

Average score	Final grade
>= 90.0	A
>= 80.0	B
>= 70.0	C
>= 60.0	D
< 60.0	F

4) Your program sorts the result based on student's name in Ascending order. Please write an output file named grades.txt. For the above input file, the output file has the following content:

name,tests,grade
Alice,4,C
Bill,4,C
Bob,2,A
Cindy,3,A
David,3,F
John,4,B

5) Grading rules – we will test your program using a different scores.txt file with different number of tests and scores.

10 points: reading file using with syntax
50 points: reading/converting scores with exception handling
30 points: calculating number of tests and grades
10 points: writing file using with syntax