Project description:
You should construct a Python 3 program containing your solution to the following problem and submit your program electronically on Moodle. The name of the file containing your code should be your student ID e.g., 12345678.py. No other method of submission is allowed. Please note that this is an individual project. Your program will be automatically run on Moodle for sample test cases provided in the project sheet, if you click the "check" link. However, your submission will be tested thoroughly for grading purposes after the due date. Remember you need to submit the program as a single file and copy-paste the same program in the provided text box. You have only one attempt to submit, therefore, do not submit until you are satisfied with your attempt. All open submissions at the time of the deadline will be automatically submitted. Once your attempt is submitted, there is no way in the system to open/reverse/modify it.

You are expected to have read and understood the University's guidelines on academic conduct. In accordance with this policy, you may discuss with other students the general principles required to understand this project, but the work you submit must be the result of your own effort. Plagiarism detection, and other systems for detecting potential malpractice, will therefore be used. Besides, if what you submit is not your own work then you will have learnt little and will therefore, likely, fail the final exam.

You must submit your project before the deadline listed above. Following UWA policy, a late penalty of 5% will be deducted for each day (or part day), after the deadline, that the assignment is submitted. No submissions will be allowed after 7 days following the deadline except approved special consideration cases.

Project Overview:
The ABC research institute collected information of different organisations from all over the world for their future investment purposes. The collected dataset contains several parameters about each organisation, such as name and id of the organisation, country of the organisation registration, category of work, foundation year, number of employees, median salary, profit in 2020 and profit in 2021.

You are required to write a Python 3 program that will read a CSV file. After reading the file, your program is required to complete the following tasks:

1) Create a dictionary and store the following information in it:
    a. t-test score of profits in 2020 and 2021 for each country.
    b. Minkowski distance between the number of employees and the median salary for each country.

2) Create a nested dictionary that contains the following information for each category of organisations.
    a) organization ID's, and a list of the following data corresponding to each organization ID:
        i. Number of employees.
        ii. Percentage of profit change from 2020 to 2021 (absolute value).
        iii. Rank of the organisation within each category, with respect to the number of employees.

Requirements:

1) You are not allowed to import any external or internal module in python. While use of many of these modules, e.g., csv or math is a perfectly sensible thing to do in production setting, it takes away much of the point of different aspects of the project, which is about getting practice opening text files, processing text file data, and use of basic Python programming skills.

2) Ensure your program does NOT call the input() function at any time. Calling the input() function will cause your program to hang, waiting for input that automated testing system will not provide (in fact, what will happen is that if the marking program detects the call(s), it will not test your code at all which may result in zero grade).

3) Your program should also not call print() function at any time except for the case of graceful termination (if needed). If your program has encountered an error state and is exiting gracefully then your program needs to return empty dictionaries and print an appropriate message. At no point should you print the program's outputs instead of (or in addition to) returning them or provide a printout of the program's progress in calculating such outputs.

Input:

Your program must define the function `main` with the following syntax:

```
def main(csvfile):
```

The input argument for this function is:

- `csvfile`: The name of the CSV file (as string) containing the record of the organisations around the world. The first row the CSV file will contain the headings of the columns. A sample CSV file "Organisations.csv" is provided with project sheet on LMS and Moodle.

Output:

Two outputs are expected:

1) A dictionary which will have country names as keys, and the corresponding value for each country (key) will be a list containing t-test score and Minkowski distance between number of employees and median salary of the respective country. The expected output is in the following format:

```
{'country1': [t-test score, minkowski distance],
'country2': [t-test score, minkowski distance],…,
 'countryn': [t-test score, minkowski distance]}
```

2) A nested dictionary 'D' which will store the different categories of organizations (such as `'transportation'`, `'apparel'`, etc.) as keys and each corresponding value will be another dictionary 'd'. Each dictionary 'd' will store the organization IDs as keys within each category of organizations and information related to the organization IDs as values. Each value of 'd' will be a list containing the following data for each organisation:
    a. number of employees,
    b. absolute percentage of profit change from 2020 to 2021, and
    c. rank of an organisation within each category with respect to the number of employees (sort them in descending order, the organisation with the higher number of employees holds the higher rank, where the highest rank is '1'). If two organizations have the same number of employees, sort them (the tied organizations' IDs only) in descending order of their profit change. Below is the format:

```
{'category1':{'organisation ID1': [number of employees,
 absolute percentage of profit change, rank ],
 'organisation ID2': [number of employees, absolute
 percentage of profit change, rank ]},
 'category2':{'organisation ID1': [number of employees,
 absolute percentage of profit change, rank ],…,
 'organisation IDN': [number of employees, absolute
 percentage of profit change, rank]},…,
 'categoryK':{'organisation ID1': [number of employees,
 absolute percentage of profit change, rank ],…,
 'organisation IDN': [number of employees, absolute
 percentage of profit change, rank]}}
```

Note: All the float results should be rounded to 4 decimal values and all the strings should be converted to lower case. Also, keep in mind that a dictionary is an *unordered* collection of key-value pairs.

Examples:
Download `Organisations.csv` file from the folder of Project 2 on LMS or Moodle. An example of how you can call your program from the Python shell (and examine the results it returns) are:

```
>>> output1, output2 = main('Organisations.csv')
```

The output variables returned are two dictionaries. Following are some examples of examining the returned dictionaries:

Example#1
```
>>> output1['brazil']
[-0.5175, 10174.3314]
```

```
>>> output2['biotechnology']
{'3c08339af3bb8c8': [8575, 36.4935, 1], 'eaf5ae0fcbcb4dd':
[6603, 78.062, 3], '139ab569bdfce4f': [3493, 62.6008, 4],
'a483cd7f7b486b4': [3427, 179.344, 5], '7ade1d82d2ac863':
[7205, 140.2845, 2], 'bf1cc30febed38c': [481, 8.9567, 6],
'bde405d2e490ebe': [92, 38.1616, 7]}
```

Example#2

```
>>> output1['afghanistan']
[0.0367, 4400.639]

>>> output2['accounting']
{'a5e8ce5cf97c2ac': [8128, 760.9484, 1], '5e2bb2dace9511e':
[7007, 73.0692, 3],'df66e70fae1aa5d': [7518, 0.5118, 2],
'795195c9db5e1c0': [6977, 96.9351, 4], 'a6bc77d5ce07c7b':
[6947, 90.4202, 5], 'b715731fa4a6cdb': [6429, 970.6279, 7],
'8f55cd0ad6dcde2': [6202, 22.6138, 8], 'ca8e1dfba7b1d8d':
[6628, 110.683, 6], 'bcaac3adb10bf1c': [6143, 801.5984, 9],
'c38cf79de2e6b6a': [5784, 125.3964, 10], 'ef56bdce48de5ff':
[5523, 597.8386, 11], '0bcebfcd12bcb7e': [5282, 31.2454, 12],
'e0da4a69658eaca': [4491, 120.9667, 13], '27fbc78271f3aa2':
[4288, 174.5934, 14], 'd457875b76d0ad8': [3784, 28.912, 15],
'ef7e820bc9f7e49': [2861, 40.1272, 16], 'a45e805db7feee1':
[2658, 158.8379, 17], 'a3b8d27d51aae2f': [2135, 64.8933, 18],
'ba907c2acbc34ba': [2090, 13.0396, 19], 'f8a35a4b5d7b2c1':
[871, 40.3551, 20]}
```

Assumptions:
Your program can assume the following:

- The order of columns can be different than the order provided in the sample file. Also, there can be extra or less columns in the testing input file. Moreover, rows can be in random order except the first row containing the headings.
- All string data in the file is case-insensitive, which means “Biotechnology” is same as “BIOTECHNOLOGY”. Your program needs to handle this situation to consider both to be the same.
- There can be missing or invalid data in a row, and in such instance(s) the entire row(s) should be ignored. Some examples of invalid data can be: negative or zero number of employees and median salary; identical organisation IDs; null/empty values in the required columns. You need to think of other invalid cases yourself.
- The necessary formulas are provided at the end of this document.

Important grading instruction:
Note that you have not been asked to write specific functions. The task has been left to you. However, it is essential that your program defines the top-level function *main(csvfile)* (hereafter referred to as *“main()"* in the project documents to save space when writing it.
Note that when *main()* is written it still implies that it is defined with its input argument). The idea is that within *main(),* the program calls the other functions. (Of course, these functions

may then call further functions.) This is important because when your code is tested on Moodle, the testing program will call your *main()* function. So if you fail to define *main(),* the testing program will not be able to test your code and your submission will be graded zero. Don't forget the submission guidelines provided at the start of this document.

Marking rubric:
Your program will be marked out of 30.

22 out of 30 marks will be awarded automatically based on how well your program completes a number of tests, reflecting normal use of the program, and also how the program handles various states including, but not limited to, different numbers of rows in the input file and / or any error states. You need to think creatively what your program may face. Your submission will be graded by data files other than the provided data file. Therefore, you need to be creative to look into corner or worst cases. I have provided few guidelines from ACS Accreditation manual at the end of the project sheet which will help you to understand the expectations.

8 out of 30 marks will be awarded on style (5/8) "the code is clear to read" and efficiency (3/8) "your program is well constructed and run efficiently". For style, think about use of comments, sensible variable names, your name at the top of the program, etc. (Please watch the lectures where this is discussed)

Style Rubric:

0	Gibberish, impossible to understand or style is poor
1-2	Style is fair
3-4	Style is good or very good, with small lapses
5	Excellent style, really easy to read and follow

Your program will be traversing text files of various sizes (possibly including large csv files) so you need to minimise the number of times your program looks at the same data items.

Efficiency rubric:

0	Code too complicated to judge efficiency or wrong problem tackled
1	Very poor efficiency, additional loops, inappropriate use of readline()
2	Acceptable or good efficiency with some lapses
3	Excellent efficiency, should have no problem on large files, etc

Automated testing is being used so that all submitted programs are being tested the same way. Sometimes it happens that there is one mistake in the program that means that no tests are passed. If the marker is able to spot the cause and fix it readily, then they are allowed to do that and your - now fixed - program will score whatever it scores from the tests, minus 4 marks, because other students will not have had the benefit of marker intervention. Still, that's way better than getting zero. On the other hand, if the bug is hard to fix, the marker needs to move on to other submissions.

Extract from Australian Computing Society Accreditation manual 2019:

As per Seoul Accord section D, a complex computing problem will normally have some or all of the following criteria:

- involves wide-ranging or conflicting technical, computing, and other issues;
- has no obvious solution, and requires conceptual thinking and innovative analysis to formulate suitable abstract models;
- a solution requires the use of in-depth computing or domain knowledge and an analytical approach that is based on well-founded principles;
- involves infrequently-encountered issues;
- is outside problems encompassed by standards and standard practice for professional computing;
- involves diverse groups of stakeholders with widely varying needs;
- has significant consequences in a range of contexts;
- is a high-level problem possibly including many component parts or sub-problems;
- identification of a requirement or the cause of a problem is ill defined or unknown.

Necessary formulas:

1) t-test calculation, t

$$t = \frac{(\bar{x}_1 - \bar{x}_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where,

$\bar{x}_1$ = Observed Mean of 1st Sample
$\bar{x}_2$ = Observed Mean of 2nd Sample
s_1 = Standard Deviation of 1st Sample
s_2= Standard Deviation of 2nd Sample
n_1 = Size of 1st Sample
n_2 = Size of 2nd Sample

2) Minkowski distance, m

$$m = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

where,

x_i = first set of data.
y_i = second set of data.
n = number of samples.
p = a value which controls the level of similarity. Use p =3 in this project.

3) Percentage

Absolute profit change for an organization:

C = |profit in 2020 – profit in 2021|

Percentage = (C / profit in 2020) * 100