IT5001 Practical Exam

- Do NOT close your Coursemology PE page browser after you open it.
    - If you accidentally close it, you need the invigilator to re-open it for you, resulting in lost time.
- You are recommended to use the Firefox browser because Chrome and Edge have slower response time with our secure network control.
- You should only log in to Coursemology with the account you are using in our module. Also, your browser should only open our PE submission page (and the question paper if you use your browser to read it).
- Only 5 submissions ("Run" Code) per part.
    - This means you can only submit/test run 5 times for each part.
    - We will NOT grant you extra test runs or "Finalize Submissions" if you accidentally use them up.
- Copy ONLY the required functions from IDLE to Coursemology.
- Do NOT submit your test cases/test code.
- You can access the help/reference from IDLE (F1 help).
- Click "Finalize Submission" to submit your exam. You will not be able to amend your code after this.
- You are **not allowed to import extra packages** other than the ones already imported in our skeleton code unless some questions let you do so specifically.
- You must name your functions exactly and use the same number of parameters as the questions stated. However, the naming for your parameters can be different.
- Submit your code earlier before the PE ends. The system may be congested at the end of the PE.
- We will only mark the code submitted on Coursemology. Any unsubmitted code, including those code you left on your computer you are using in the PE, will be given zero mark and it is your own responsibility to submit your code correctly.
- The whole assessment is divided into a few parts. If you reuse a function from a previous part, you must copy it over to the current part on Coursemology.
- You are allowed to write extra helper functions to structure your code better. However, remember to submit them together mentioned in the point above.
- In all parts, you should **return** values instead of printing your output. In another word, you should not need any "`print()`" in this entire PE for submission unless you are told to "`print`" specifically.

Part 1 Fizz Buzz (30 marks)

Fizz buzz is a group word game for children to teach them about division. However, before we get too advanced into both Fizz and buzz, let's just do a Fizz game.
In the Fizz game, a child counts incrementally, replacing any number divisible by a factor X with the word "Fizz". For example, if X = 4, and the child will count as follows:

```
1, 2, 3, 'Fizz', 5, 6, 7, 'Fizz', 9, 10, …
```

In this part, you can assume all the factor(s) is/are integers >= 1.

Task 1 The Fizz game Iterative version (7 marks)

Write an ***iterative*** version of the `fizz_i(factor1,n)` to **return** a list with integers and the string 'Fizz' as mentioned above. The list contains integers from 1 to `n` with `n` >= 0. However, the string 'Fizz' will replace any integer that is divisible by `factor1`. Your function cannot be recursive in this task.

```
>>> print(fizz_i(4,10))
[1, 2, 3, 'Fizz', 5, 6, 7, 'Fizz', 9, 10]
>>> print(fizz_i(3,8))
[1, 2, 'Fizz', 4, 5, 'Fizz', 7, 8]
>>> print(fizz_i(7,15))
[1, 2, 3, 4, 5, 6, 'Fizz', 8, 9, 10, 11, 12, 13, 'Fizz', 15]
>>> print(fizz_i(1,5))
['Fizz', 'Fizz', 'Fizz', 'Fizz', 'Fizz']
>>> print(fizz_i(4,0))
[]
```

Task 2 The Fizz game Recursive version (7 marks)

Write a ***recursive*** version of the function `fizz_r(factor1,n)` with the same functionality in Part 1 Task 1. However, you cannot use any loops or list comprehension in this task.

Task 3 The Fizz game lambda version (7 marks)

Write a ***lambda*** version of the function `fizz_l(factor1,n)` with the same functionality in Part 1 Task 1. Your function should be **one line only** (without any semicolon `';'`) in a form like this:

```
fizz_l = lambda factor1,n: #your code here
```

Task 4 The Fizz Buzz game (9 marks)

The real Fizz Buzz game is a game with two *different* factors X and Y. An integer will be replaced by:

- 'Fizz' if it's divisible by X but not by Y, or
- 'Buzz' if it's divisible by Y but not by X, or
- 'FizzBuzz' if it's divisible by both X and Y.

You can implement the function in either iteration, recursion, or any form you want. Here is some sample output:

```
>>> print(fizzbuzz(2,3,10))
[1, 'Fizz', 'Buzz', 'Fizz', 5, 'FizzBuzz', 7, 'Fizz', 'Buzz', 'Fizz']
>>> print(fizzbuzz(3,4,13))
[1, 2, 'Fizz', 'Buzz', 5, 'Fizz', 7, 'Buzz', 'Fizz', 10, 11, 'FizzBuzz', 13]
```

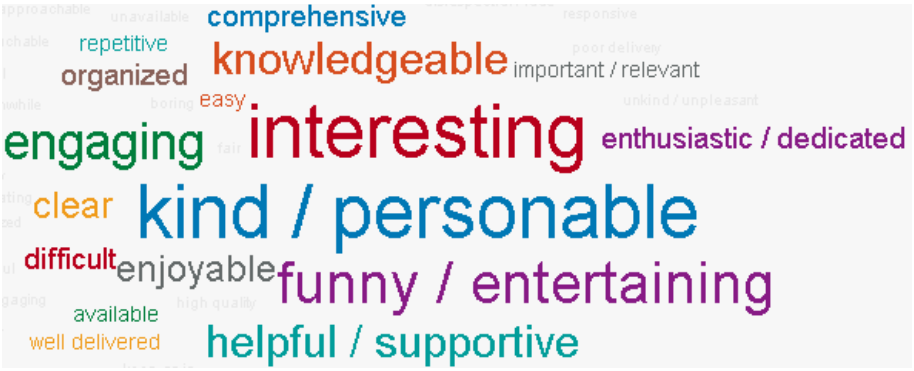
Part 2 Text Analytics (30 marks)

Each year we lecturers will receive a lot of students' feedback on our teaching. With hundreds of students, the feedbacks are very long:

Comment
- Prof Cheng showed his care for all the students. He always gives encouragement and advices to help us pass this difficult module.
- Very organized and structured way of teaching.
- not only did he teach us coding but also life lessons and linked coding to other stuff
- funny and creative and smart
- ok
- I like his teaching style and sense of humor
- Professor Alan shows his expertise and professionalism not only in the academic side, but also in the emotional side. When you listen to his lecture, you learn from his teaching and his attention to his students' understanding helps elevate that
- Prof Alan encourages students not to give up after a setback
- He is very engaging
- Helped me appreciate computer science much more, and made me want to improve my skills. I think I would not be particularly interested in computer science otherwise.
- Hardworking professor. Spends times to ensure that his students understand the concept before moving on to the next.
- Prof Alan is dedicated and he always comes up with interesting and fun ways to engage us during lectures.
- hes a great prof
- memes
- Always motivating his students and give prep talks before major tests.
- appreciate how he always tries to teach us how to think in life in general. i enjoy his lectures very much and he never fails to try his best to help the students with our questions
- Fun and approachable teacher. Willing to go out of way to help and answer doubts despite end of lecture
- Very passionate about imparting computer science knowledge to us. Aside from cs, we were taught on the growth mindset and how to become a better programmer and person.
- Great Teacher
- made cs interesting

Continue for many pages…

With text analytics, we can extract what are the meaning, insight or extract out of some long text. Here is the text analytics report of the most frequently mentioned words:



And the most frequent words are

Attributes [No. of comments]	Overall [35]
KIND / PERSONABLE	23 %
INTERESTING	23 %
FUNNY / ENTERTAINING	17 %
ENGAGING	14 %
KNOWLEDGEABLE	11 %

In this part, your task is to generate such report with any text.

Part 2 Task 1 Cleaning the Text

Given any text file, e.g. the file of the first paragraph of the Gruffalo Story (‘`gruffalo first para.txt`’).

```
A mouse took a stroll through the deep dark wood. A fox saw the mouse, and the mouse looked good.
"Where are you going to, little brown mouse? Come and have lunch in my underground house."
"It's terribly kind of you, Fox, but no - I'm going to have lunch with a gruffalo."
"A gruffalo? What's a gruffalo?"
"A gruffalo! Why, didn't you know?
He has terrible tusks, and terrible claws, And terrible teeth in his terrible jaws."
```

The first step is to

- Convert every alphabet into lower case (the underlined characters above)
- Keeping only the alphabets from ‘a’ to ‘z’. Namely, changing all the other characters to spaces

For example the last line of the above text will become:

```
a mouse took a stroll through the deep dark wood  a fox saw the mouse  and the mouse looked good
 where are you going to  little brown mouse  come and have lunch in my underground house
 it s terribly kind of you  fox  but no   i m going to have lunch with a gruffalo
 a gruffalo  what s a gruffalo
 a gruffalo  why  didn t you know
he has terrible tusks  and terrible claws  and terrible teeth in his terrible jaws
```

Your **task** here is to write a function `clean_text(filename,minlen)` to read in a text file named `filename`, and return a **`list`** of all the words broken down by the above method without spaces. However, you will ***remove*** any word that has a length <= `minlen`. (Actually the minimum length of words is `minlen + 1`.)

```
>>> print(clean_text('gruffalo first para.txt',4))
['mouse', 'stroll', 'through', 'mouse', 'mouse', 'looked', 'where', 'going', 'little', 'brown', 'mouse',
'lunch', 'underground', 'house', 'terribly', 'going', 'lunch', 'gruffalo', 'gruffalo', 'gruffalo',
'gruffalo', 'terrible', 'tusks', 'terrible', 'claws', 'terrible', 'teeth', 'terrible']
>>> print(clean_text('gruffalo first para.txt',5))
['stroll', 'through', 'looked', 'little', 'underground', 'terribly', 'gruffalo', 'gruffalo', 'gruffalo',
'gruffalo', 'terrible', 'terrible', 'terrible', 'terrible']
```

Finally, the order in your output list must be in the same order of appearance in the original text file. We provided a few more text files for you to try out.

Part 2 Task 2 Text Analytics

Write a function `text_analytics(clean_text_list,n)` takes in a list of words `clean_text_list` from Task 1, and an integer `n`, and return a dictionary with the n highest frequencies with the words. And print all if there less than n of them. For example, in the file above, the three words 'terrible', 'gruffalo', 'mouse' appear most with a frequency of 4, and the three words 'going', 'have', 'lunch' appear second most with a frequency of 2, therefore, if we want to output the 2 highest frequencies words, it will be:

```
>>> gruf1 = clean_text('gruffalo first para.txt',3)
>>> print(text_analytics(gruf1,2))
{4: {'terrible', 'gruffalo', 'mouse'}, 2: {'going', 'have', 'lunch'}}
```

For the other file of the whole story of the Gruffalo (‘`gruffalo.txt`’), here is the sample output:

```
>>> gruf = clean_text('gruffalo.txt',4)
>>> print(text_analytics(gruf,5))
{30: {'gruffalo'}, 26: {'mouse'}, 9: {'snake', 'little'}, 7: {'terrible'}, 6: {'goodbye', 'house',
'where'}}
```

Hint: You can use `pprint` to print a dictionary in a better way, e.g. `pprint(your_dict,sort_dicts=False)`.

Part 3 Red Blood Cells Study (40 marks)

We got some photos of red blood cells and we want to study them. Here are the two things we want to find out:

- With two photos, what are the differences between them
- Given one photo, count how many cells are there

A red blood cell photo is digitized into a 2D array. For each entry of the array is either a period '`.`' indicating a part of a red blood cell, or a pound sign '`#`' indicating an empty space like the following:

```
##############
##..#########
#....########
##..###.#####
######...####
```

The above picture shows two red blood cells. Notice that the cells are in different sizes. The photos are given to you with two different formats

- String Representation: A 1D list with each entry is a string of each row of the image, or
- 2D Representation: A 2D array with each entry is one of the pixel of the picture.

For example, the above picture can be represented by the following two representations:

String Representation	**2D Representation**

`['##############',`
` '##..#########',`
` '#....########',`
` '##..###.#####',`
` '######...####']`
	`[['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#'],`
	` ['#', '#', '.', '.', '#', '#', '#', '#', '#', '#', '#', '#', '#'],`
	` ['#', '.', '.', '.', '.', '#', '#', '#', '#', '#', '#', '#', '#'],`
	` ['#', '#', '.', '.', '#', '#', '#', '.', '#', '#', '#', '#', '#'],`
	` ['#', '#', '#', '#', '#', '#', '.', '.', '.', '#', '#', '#', '#']]`

Part 3 Task 1 Format Conversion

Write two functions:

- `str_to_two_d(pic)`: return the 2D representation of `pic` that is in the string representation
- `two_d_to_str(pic)`: return the string representation of `pic` that is in the 2D representation

```
>>> pic = ['##############',
 '##..#########',
 '#....########',
 '##..###.#####',
 '######...####']
>>> pprint(two_d_to_str(str_to_two_d(pic)))
['##############',
 '##..#########',
 '#....########',
 '##..###.#####',
 '######...####']
>>> pprint(str_to_two_d(pic))
[['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#'],
 ['#', '#', '.', '.', '#', '#', '#', '#', '#', '#', '#', '#', '#'],
 ['#', '.', '.', '.', '.', '#', '#', '#', '#', '#', '#', '#', '#'],
 ['#', '#', '.', '.', '#', '#', '#', '.', '#', '#', '#', '#', '#'],
 ['#', '#', '#', '#', '#', '#', '.', '.', '.', '#', '#', '#', '#']]
```

Part 3 Task 2 Compare Two Pictures with the Same Size

Given two pictures in the same size, namely, the same number of row and same number of columns, we want to compute the difference between the two pictures.

Write a function `pic_diff(pic1,pic2)` to take in two pictures (in either String or 2D representations), and return the String representation of the picture with the following rule:

- If the pixels of the two pictures are the same, put an equal sign '='
- If the pixels of the two pictures are different, put a capital letter 'X'.

For example, here are two pictures in String representations (the character in `pic2` are bolded and colored red to indicate the difference.):

pic1	pic2
['##########', '##..######', '#....#####', '#....##.##', '##..##...#', '#######.##']	['##########', '##..######', '#....#####', '#**#**..**#**##.##', '##**##**##...#', '#######.##']

Then the output should be:

```
>>> pprint(pic_diff(pic1,pic2))
['==========',
 '==========',
 '==========',
 '=X==X=====',
 '==XX======',
 '==========']
```

Part 3 Task 3 Compare Two Pictures with the Different Sizes

In this task, you do NOT need to submit another function but just to add this functionality to the function you wrote in Task 2. However, be careful because you do not want to ruin your result for Task 2.

The two input pictures could be in different sizes. Let's assume pic1 has r1 rows and c1 columns, and pic2 has r2 rows and c2 columns. Then you should return a picture in string format with max(r1,r2) rows and max(c1,c2) columns. In the picture, you should show the difference of the pixel same as Task 1 if the pixel is in both pic1 and pic2. However

- If the pixel is in pic1 but not in pic2, put a string '1' in the output
- If the pixel is in pic2 but not in pic1, put a string '2' in the output
- If the pixel is neither in pic1 nor in pic2, put a string '0' in the output

For example here are two picture with different sizes

pic1 (6 rows and 10 columns)	pic2 (5 rows and 13 columns)
['##########', '##..######', '#....#####', '#....##.##', '##..##...#', '#######.##']	['#############', '##..#########', '#....########', '##..###.#####', '######...####']

Your output should be a picture of 6 rows and 13 columns like the following:

```
>>> pprint(pic_diff(pic1,pic2))
['==========222',
 '==========222',
 '==========222',
 '=X==X=====222',
 '==XX======222',
 '1111111111000']
```

You should not write another function. You should just add in this functionality into your function in Task 2.

Part 3 Task 4 Counting the Number of Red Blood Cells

Given a picture in either representation, write a function `count_cells(pic)` to return the number of red blood cells in integer. The characters '.' in the picture are "connected" to each other as the same red blood cell if they are next to each other in the north, south, east or west directions only. However, two '.' are not considered as connecting if they are connecting diagonally. Please see the following example for illustration.

```
>>> pic3 = ['..###########',
            '##..#####..##',
            '#....###....#',
            '##..###.#..##',
            '######...####',
            '######...####',
            '#######.#####']
>>> print(count_cells(pic3))
4
```

-- End of Paper --