Module Code
COM00015C

UNIVERSITY of York

BEng, BSc, MEng and MMath Degree Examinations

Department Computer Science

Title Software 1: Mock Exam

Time Allowed FIVE hours

> Papers late by up to 30 minutes will be subject to a 5 mark penalty; papers later than 30 minutes will receive 0 marks.
>
> The time allowed includes the time to download the paper and to upload the answers.

Word Limit Not Applicable

Allocation of Marks: Question 1 is worth 10%, question 2 and question 3 are worth 15%, and question 4 is worth 40%, adding up to a total of 80%.

Instructions:

> Candidates must answer **all** questions using Python 3.10 or above. Failing to do so may result in a mark of 0%. All questions are independent and can be answered in any order.
>
> Download the paper and the required source files from the VLE, in the "Assessment>SOF1 2023-24 mock exam" section. Once downloaded, unzip the file. You **must** save all your code in the files provided. **Do not** save your code anywhere else other than this folder. Submit your answers to the GradeScope submission point named **SOF1 mock exam**. You can find the GradeScope submission point on the "Assessment" page on the VLE.

Note on Academic Integrity

We are treating this online examination as a time-limited open assessment, and you are therefore permitted to refer to written and online materials to aid you in your answers. However, you must ensure that the work you submit is entirely your own, and for the whole time the assessment is live you must not:

- communicate with other students on the topic of this assessment.
- communicate with departmental staff on the topic of the assessment other than to highlight an error or issue with the assessment which needs amendment or clarification).
- seek assistance with the assessment from academic support services, such as the Writing and Language Skills Centre or Maths Skills Centre, or from Disability Services (unless you have been recommended an exam support worker in a Student Support Plan).
- seek advice or contribution from any other third party, including proofreaders, friends, or family members.

We expect, and trust, that all our students will seek to maintain the integrity of the assessment, and of their award, through ensuring that these instructions are strictly followed. Where evidence of academic misconduct is evident this will be addressed in line with the Academic Misconduct Policy and if proven be penalised in line with the appropriate penalty table. Given the nature of these assessments, any collusion identified will normally be treated as cheating/breach of assessment regulations and penalised using the appropriate penalty table (see AM3.3. of the Guide to Assessment).

1 (10 marks) Basic Programming Structure

The code must be written in the provided file `question_1.py`.

When you start to learn touch typing, you only practice on a subset of keys on the keyboard. Quite often the exercises consist of repeating sequences of this subset of keys, however, most of the time these sequences do not have a meaning or do not even represent an existing word. In order to make the exercises more interesting and engaging, we want to take a text from a book, and keep only the words from that text that contains only letters from a subset of keys.

Write a function `extractText(text, keys)` that returns a string containing only words from the string `text` that are only composed of letters from the string `keys`.

For simplicity we make the following assumptions:

- The string `text` contains only alphabet letters and blank spaces, no numbers or punctuation,
- the string `keys` is not empty,
- the parameters provided will satisfy the two previous statement, and there is no need to check the inputs.

In addition, the function must meet the following requirements:

- each word in the returned string is separated by a single blank space,
- the returned string does not start or end with a blank space,
- the function should be case insensitive, that is the function should return the string `'Reader'` if the input text is `'Reader'` and the keys parameter is `'ArEdz'`,
- if the parameter `text` is an empty string, the function returns an empty string.

For example:

```
>>> data = 'The term conda is not recognised as the name of a'
>>> extractText(data, 'theAsORin')
'The is not as the a'
>>> extractText('', 'theAsORin')
''
```

2 (15 marks) List Built-in Data Structure

The code must be written in the provided file `question_2.py`.

To perform an efficient operation on objects in a scene, an object is sometime approximated by a sphere-tree as shown in Figure 1. For simplicity, we will represent an object as a set of spheres rather than a sphere-tree. A sphere S_i can be defined by three values (x_i, y_i, z_i, r_i) where x_i, y_i and z_i are the coordinates of the centre of the sphere, and the value r_i is the radius of the sphere.

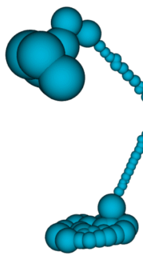


Figure 1: An example of a lamp approximation using spheres.

In order to build complex objects we can use the union of multiple (simpler) objects. The aim of this question is to take two objects, each of them represented by a set of spheres, and create the union of the two objects by using the union of the two sets of spheres.

The simplest way to do the operation is to take all the spheres of `objectA` and all the spheres of `objectB` and add them to the set of spheres of the resulting object. However, by doing so we may have some redundant information, that is in the resulting union we may have a sphere S_1 from `objectA` comprised within a sphere S_2 from `objectB` (or vice versa). This is not satisfactory, therefore the solution should remove from the set any sphere that is comprised within another one. A sphere S_1 lies within a sphere S_2 if and only if:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} + r_1 \leq r_2$$

(i) [15 marks] Write a function `union(objectA, objectB)` that returns a set of spheres (set of tuples `(x, y, z, radius)`) representing the union of the two objects. The parameters `objectA` and `objectB` are sets of tuples, where the tuples are of the form `(x, y, z, radius)`. The values `x`, `y` and `z` are the coordinates of the centre of a sphere, and `radius` is the radius of a sphere. The returned set must not contain any spheres that are comprised within another sphere. For simplicity, we assume that the inputs provided are correct, that is all tuples contain exactly 4 floats and the radius given in each tuple is strictly positive.

3 (15 marks) Python I/O

The code must be written in the provided file `question_3.py`.

In American football, passer rating is a measure of the performance of passers, primarily quarterbacks. Passer rating is calculated using a player's passing attempts, completions, yards, touchdowns, and interceptions. The NFL passer rating formula includes four variables:

- completion percentage:
$$a = \left(\frac{COMP}{ATT} - 0.3\right) \times 5$$
- yards per attempt:
$$b = \left(\frac{YDS}{ATT} - 3\right) \times 0.25$$
- touchdowns per attempt:
$$c = \frac{TD}{ATT} \times 20$$
- interceptions per attempt:
$$d = 2.375 - \left(\frac{INT}{ATT} \times 25\right)$$

Where ATT is the number of passing attempts, $COMP$ is the number of completions, YDS is the passing yards, TD is the number of Touchdown passes, and INT is the number of interceptions. Each of those variables is scaled to a value between 0 and 2.375, that is if the result of any calculation is greater than 2.375, it is capped to 2.375, and if the result is a negative number, it is set to zero. Then, the four variables are used to calculate the passer rating:
$$\text{Passer Rating} = \left(\frac{a+b+c+d}{6}\right) \times 100$$

The aim of this question is to read a CSV file containing the stats of NFL quarterbacks and return a dictionary containing the name of the quarterbacks as keys and their passer ratings rounded to 1 decimal place as values. The file format is as follows:

- lines starting with # should be ignored,
- each line contains the data of a single player,
- the entries on each line are as follow:
player name, attempts, completions, yards, Touch Downs, and interceptions in that order, and each entry is separated by a comma.

Implement a function `getPasserRating(filename)` that reads the CSV file `filename` containing the players statistics using the format described earlier, and returns a dictionary where the keys are the names of the players and the values are their passer ratings rounded to 1 decimal place.

- The function must raise a `ValueError` if the file does not follow the format described earlier.
- The function must raise a `KeyError` if the file contains more than one data set for a given player, that is two lines or more contain the same player's name.

For example, given the CSV file containing the following:

```
#player name,attempts,completion,yards,Touch Down,interception
Justin Herbert,166,111,1250,9,2
Josh Allen,168,113,1227,10,3
```

The returned dictionary should be:

```
{'Justin Herbert': 102.2, 'Josh Allen': 101.0}
```

4 (40 marks) User Defined Data Structure

The code must be written in the provided file `question_4.py`.

The aim of this question is to solve a sliding puzzle game. For simplicity, the size of the puzzle is reduced to a 4×4 board containing 15 tiles and an empty space. The aim of the game is to move (slide) the tiles until the centre of the board matches a given pattern as shown in Figure 2.

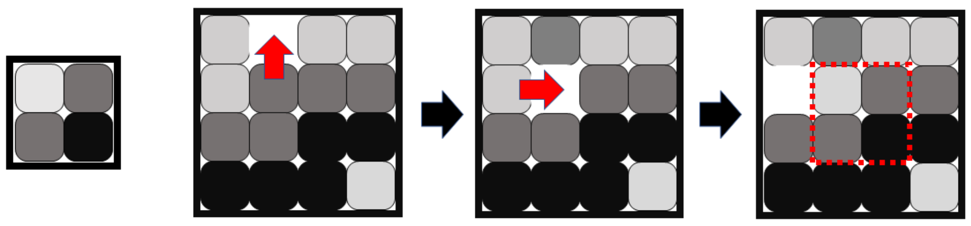
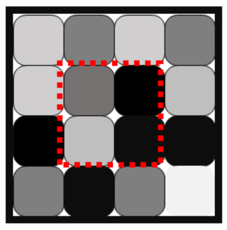


Figure 2: An example of a pattern (left) and the two moves needed to solve the puzzle.

You will implement a class `ColourPuzzle` to simulate the game. We made the design decision to represent the board by a 2D list of `int`, where the three colours used are numbered 1 to 3 and the empty space is represented by 0 as shown in Figure 3.



```
[[2, 1, 2, 1],
 [2, 1, 3, 2],
 [3, 2, 3, 3],
 [1, 3, 1, 0]]
```

Figure 3: An example of a puzzle (left) and its representation as a 4×4 2D list of int.

(i) [15 marks] The class `ColourPuzzle` has a single **protected** instance attribute `_board`. The attribute `_board` is a $2D$ `list` of `int` representing the state of the puzzle. The representation of the puzzle is shown in Figure 3.

Implement the `__init__` method having a single parameter `puzzle`. The parameter `puzzle` is a $2D$ `list` of `int` representing the initial state of the puzzle.

The method should initialise the instance attribute `_board` with a **copy** of the parameter `puzzle`. In addition, the method must raise a `ValueError` if the parameter `puzzle` is not a valid puzzle, that is:

- it is not a 4×4 2D list of int,
- it does not contain 5 tiles of each colour $\{1, 2, 3\}$ and one empty space $\{0\}$.

(ii) [5 marks] Within the class `ColourPuzzle`, implement the method `matchPattern` that takes a single parameter `pattern`, a 2×2 2D list of `int` representing a pattern to be matched by the puzzle. For simplicity, we assume that the object passed in the parameters is a valid pattern, that is:

- the object is a 2×2 2D list of `int`,
- the elements in the 2D list are only valid colours (values are in the set $\{1, 2, 3\}$).

The method returns `True` if the centre of the puzzle matches the pattern, `False` otherwise. For example, considering the pattern and the puzzle's states given in Figure 2, the method returns `False` for the puzzle on the left-hand side and `True` for the puzzle on the right-hand side.

(iii) [10 marks] Within the class `ColourPuzzle`, implement the following methods that take no parameters, moves a tile to the empty space and returns `True` if the move is possible, `False` otherwise.

- `moveLowerTile` moves the tile below the empty space up if the empty space is not in the bottom row of the puzzle,
- `moveLeftTile` moves the tile on the left of the empty space to the right if the empty space is not in the leftmost column of the puzzle,
- `moveUpperTile` moves the tile above the empty space down if the empty space is not in the top row of the puzzle,
- `moveRightTile` moves the tile on the right of the empty space to the left if the empty space is not in the rightmost column of the puzzle.

If the move is impossible, the puzzle must not be modified. Figure 4 shows the four moves. When considering the second from the right puzzle in Figure 4, calling `moveUpperTile` or `moveLeftTile` must return false and the puzzle must remain unchanged as thse are not possible moves.

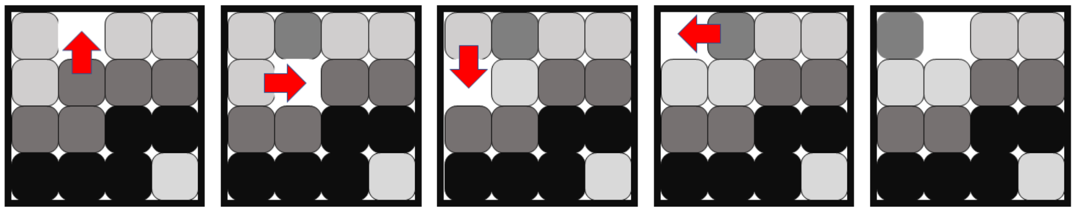


Figure 4: An example of the four moves available to play. From left to right: $moveLowerTile$, $moveLeftTile$, $moveUpperTile$, and $moveRightTile$. Note also that in the most right puzzle, the $moveUpperTile$ should return false and the puzzle should remain unchanged as it is not a possible move.

(iv) [10 marks] Implement the method `solvable` that takes two parameters, the first parameter `pattern` is a 2×2 2D list of `int` representing a pattern to be matched, and the second parameter n is a positive integer representing a number of moves. The method returns `True` if the puzzle can be solved in at most n moves, `False` otherwise.

For simplicity, we assume that the inputs provided in the parameters are valid, that is $n \geq 0$ and `pattern` is a 2×2 2D list of `int` only containing the values in the set $\{1, 2, 3\}$.

For example, considering the leftmost puzzle and the pattern shown in Figure 2, the method should return:

- `False` if $n < 2$,
- `True` if $n \geq 2$.

Be careful, we are not looking at the minimal number of moves to solve the puzzle, and we do not records the moves needed to solve it either. The method simply answers the question "Can I solve the puzzle in n or less moves?".

End of examination paper