Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code:	CMT120
Module Title:	Fundamentals of Programming
Lecturers:	Federico Liberatore, Martin Chorley, Natasha Edwards
Assessment Title:	Programming Challenges
Date Set:	31st October 2022
Submission date and Time:	12th December 2022 at 9:30AM
Feedback Return Date:	30th January 2023

If you have been granted an extension for extenuating circumstances, then the submission deadline and return date will be 1 week later than that stated above.

If you have been granted a deferral for Extenuating Circumstances, then you will be assessed in the summer resit period (assuming all other constraints are met).

This assignment is worth 40% of the total marks available for this module. If coursework is submitted late (and where there are no extenuating circumstances):

1. If the assessment is submitted no later than 24 hours after the deadline, the mark for the assessment will be capped at the minimum pass mark;

2. If the assessment is submitted more than 24 hours after the deadline, a mark of 0 will be given for the assessment.

Extensions to the coursework submission date can ***only*** be requested using the Extenuating Circumstances procedure. Only students with approved extenuating circumstances may use the extenuating circumstances submission deadline. Any coursework submitted after the initial submission deadline without ***approved*** extenuating circumstances will be treated as late.

More information on the extenuating circumstances procedure can be found on the Intranet: `https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/extenuating-circumstances`

By submitting this assignment you are accepting the terms of the following declaration:

I hereby declare that my submission (or my contribution to it in the case of group submissions) is all my own work, that it has not previously been submitted for assessment and that I have not knowingly allowed it to be copied by another student. I understand that deceiving or attempting to deceive examiners by passing off the work of another writer, as one's own is plagiarism. I also understand that plagiarising another's work or knowingly allowing another student to plagiarise from my work is against the University regulations and that doing so will result in loss of marks and possible disciplinary proceedings[1].

Assignment

To complete this coursework, you must complete a set of programming challenges in Python and JavaScript.

Each challenge can be awarded a maximum of 10 marks. Therefore, perfectly solving five exercises will give you 50 marks (pass), and perfectly solving all the exercises will give you 100 marks. An exercise is solved perfectly only if high-quality functional code is submitted in both Python and JavaScript. Providing high-quality functional code in only one programming language results in a lower mark (i.e., five marks out of ten). Therefore, you can still pass the coursework by only completing problems in one language, or by completing half the problems in both languages.

You might not be able to solve all the exercises. *This is fine.* You are not all expected to be able to solve all of the challenges in both languages. However, you should be able to solve enough of the exercises in one or both languages to be able to pass the assessment and demonstrate you have met the learning outcomes being assessed.

The challenges are described in detail below, and you are also provided with a set of test cases that will check whether your code produces the required output or not. In particular, you will be given two test cases per exercise. You should make sure that your submitted code passes the supplied tests to ensure it functions correctly. However, please note that your code

[1] `https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/academic-integrity/cheating-and-academic-misconduct`

will be tested against a further two different test cases, which you have not been supplied with. In total then each exercise will be tested against four test cases, including the two provided. You should therefore ensure that you try to cover all possible inputs and that your code still functions correctly. Your code will need to pass all 4 tests (2 seen, 2 unseen) in order to score full marks for the functionality.

Instructions for completing the challenges

- You will find template code for the assignment on Learning Central. This provides two folders, `python` and `js`. Inside each folder you will find a `template.{js/py}` file, in which you should complete your solutions. You will also find a `test_template.{js/py}` file containing the test cases that will check your code's functionality, along with a folder of test data required for some of the tests. You are also supplied with a `Readme.md` file containing detailed instructions on how to run the test cases to check your code.

- In the templates, the functions' interfaces are given but the functions' bodies are empty. Solve the exercises by correctly filling in the functions' bodies.

- It is forbidden to change the functions' interfaces. However, new functions can be defined to support the solution of the exercises. These functions must have names that are different from those already present in the templates.

- You are NOT allowed to import any additional modules. Use of module functions will result in zero marks for the corresponding exercises.

- In all the exercises, you can assume that the inputs are provided in the appropriate format and type. Therefore, error-checking is not needed.

- The final submission should NOT contain any `input`, `print`, or `console.log` statements.

You will be given marks for solving each problem in both programming languages. Further marks will be awarded for solution style and quality. The mark scheme is described in further detail later.

Exercise 1: Iris Species Classifier

The diagram in Figure 1 shows how to classify an iris flower into one of three species (i.e., setosa, versicolor and virginica) according to its characteristics: sepal length (`Sepal.Le`), sepal width (`Sepal.Wi`), petal length (`Petal.Le`), and petal width (`Petal.Wi`).

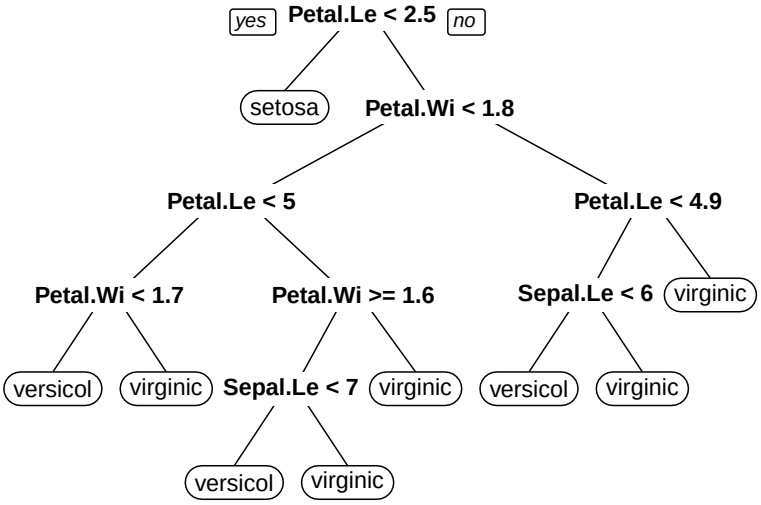

Figure 1: Iris decision tree.

Complete function `exercise1(SepalLen,SepalWid,PetalLen,PetalWid)`, taking as input four floats representing an iris flower characteristics and returning a string containing the name of the corresponding species, i.e., setosa, versicolor or virginica.

Examples:

- `exercise1(1.5,0.7,2,2.3)` returns '`setosa`'.
- `exercise1(1.9,1.5,2.7,2.5)` returns '`versicolor`'.

Exercise 2: Dog Breeds Standards

Dog breeds have different standards. Write a function that, given the breed, the height (in inches), the weight (in pounds) and the sex of a dog, returns `True`/`true` if the dog complies with the breed standard, and `False`/`false` otherwise. In particular, for the sake of this exercise, a dog complies with the standard if it is within 10% (inclusive) of the average height and weight for its breed and sex. The list of breeds and standard characteristics considered

in this exercises is given in Table 1 (please, note that all the numbers are made up).

Breed	Male height	Male weight	Female height	Female weight
Bulldog	15	50	14	40
Dalmatian	24	70	19	45
Maltese	9	7	7	6

Table 1: Exercise 2. List of breeds and standard characteristics.

Complete function `exercise2(breed,height,weight,male)`, taking as input a string, `breed`, two floats, `height` and `weight`, and a bool, `male`, taking value `True`/`true` if the dog is male and `False`/`false` otherwise. The function should return `True`/`true` or `False`/`false` depending on whether the dog complies with the breed standard, according to the rule given above.

Examples:

- `exercise2('Maltese',9.5,6.7,True)` should return `True`, as the dog complies with the standard for the Maltese breed.
- `exercise2('Bulldog',16,44,False)` should return `False`, as the dog does not comply with the standard for the Bulldog breed (it is too tall).

Exercise 3: Basic Statistics

The function in this exercise is given a list/array of floats. The function should return a list of two tuples (python) or an array of two arrays (javascript). The first returned list/array contains the minimum, the average, the median and the maximum values for the input list/array (the average and the median are rounded to the second decimal digit). The second returned list/array contains the same statistics, this time calculated from the square of the input list/array values.

The *average* is defined as the sum of the values, divided by the number of values (e.g., the average of `[1,5,4,6]` is `4`).

The *median* is defined as the "middle value" of a list of ordered numbers (note that the values in the input list might not be sorted). If the list has an odd number of elements, then the median equals the value in the middle (e.g., the median of `[1,2,3]` is `2`). If the list has an even number of elements, then the median is equal to the average between the two central values (e.g., the median of `[1,2,3,4]` is `(2+3)/2 = 2.5`.

Complete function `exercise3(l)` that takes a list/array of numbers `l` and returns a list of two tuples (python) or an array of two arrays (javascript) containing summary statistics, as illustrated above.

Examples:

- `exercise3([1,2,3,4,5])` should return `[(1,3,3,5),(1,11,9,25)]`.
- `exercise3([7,2,4,5])` should return `[(2,4.5,4.5,7),(4,23.5,20.5,49)]`

Exercise 4: Finite-State Machine Simulator

A finite-state machine (FSM) or finite-state automaton (FSA, plural: automata), finite automaton, or simply a state machine, is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition. An FSM is defined by a list of its states, its initial state, and the inputs that trigger each transition (Wikipedia contributors, 2022a).

Complete the function `exercise4(trans,init_state,input_list)`. `trans` is the dictionary describing the FSM, where the keys are `"state/input"` strings (i.e., current state and input, respectively) and the values are `"state/output"` strings (i.e., next state and output, respectively). `init_state` is the initial state. `input_list` is a list of input values. The function returns the list of outputs. You can assume that all the inputs, the outputs and the states are strings.

Examples:

- Let `trans` be `{"a/0":"a/1", "a/1":"a/0"}`, `init` be `'a'`, and `input` be `['0','0','1','1','0','0']`. The corresponding output is `['1','1','0','0','1','1']`.
- Let `trans` be `{"a/0":"a/1", "a/1":"b/0","b/0":"b/0","b/1":"a/1"}`, `init` be `'a'`, and `input` be `['0','0','1','1','0','0']`. The corresponding output is `['1','1','0','1','1','1']`.

Exercise 5: Document Stats

Write function `exercise5(filename)` that reads a text file and provides as output a tuple (python) or an array (javascript) containing the following values in the following order:

- the number of letters in the file
- the number of numeric characters in the file
- the number of symbol characters in the file (i.e., characters that are not alphanumeric and are not whitespaces)

- the number of words in the file (where we assume that words are only made of alphanumeric characters, and are separated by whitespace or punctuation)
- number of sentences in the file (you can assume that a sentence ends with a dot, question mark, or exclamation mark)
- number of paragraphs in the file (two paragraphs are separated by an empty line).

Example: Suppose the text file contains the following text:

```
She sells 10 sea
shells by
the 7-seas' shores.


The 10 shells she sells
are surely
seashells.So if she
sells 7 shells

on the 7-seashores,I'm
sure she sells
7-seashores' shells.
```

Then, the function should return `(128, 8, 10, 36, 3, 3)`.

Exercise 6: List Depth

Complete the function `exercise6(l)` that given a list or array `l`, calculates the maximum depth of the list/array. The maximum depth of a list/array without sub-lists/sub-arrays is 1. Otherwise, the maximum is one more than the maximum depth of its sub-lists/sub-arrays.

Examples:

- `exercise6([1,2,3])` returns `1`.
- `exercise6([1,[2,[]],[4,5]])` returns `3`.

Exercise 7: Change, please

Write a function that determines if it is possible to use a specific number of coins (£2, £1, 50p, 20p, 10p, 5p, 2p, and 1p) to obtain a specific total. For example, it is possible to have a total of £1 using five coins if they are all 20p. However, there is no way to have a total of £1 using 3 coins.

Complete function `exercise7(amount,coins)` that returns `True`/`true` if the total `amount` can be obtained using exactly the specified number of `coins`, or `False`/`false` otherwise.

Examples:

- `exercise7(3,2)` returns `True`, as one £2 coin and one £1 coin equal £3.
- `exercise7(5,2)` returns `False`, as £5 cannot be totalled using only two coins.

Exercise 8: Five Letter Unscramble

For this exercise you need to use the provided `wordle.txt` file that contains a list of 5-letter words used in the game Wordle (Disclaimer: This list is taken directly from the Wordle game and therefore may contain some words which some people may find offensive. Reader discretion is advised).

Complete function `exercise8(s)` that, given a string containing letters, returns the number of unique words in `wordle.txt` that can be obtained by rearranging the characters in the string. Each character in `s` can be used at most once.

Examples:

- `exercise8('sehuoh')` returns 1, as the string can be rearranged into `'house'`.
- `exercise8('caarto')` returns 5, as the string can be rearranged into `'carat'`, `'carta'`, `'actor'`, `'aorta'`, `'taroc'`.

Exercise 9: Wordle Set

For this exercise you need to use the provided `wordle.txt` file that contains a list of 5-letter words used in the game Wordle (Disclaimer: This list is taken directly from the Wordle game and therefore may contain some words which some people may find offensive. Reader discretion is advised).

Wordle[2] is a web-based word game created and developed by Welsh software engineer Josh Wardle. Players have six attempts to guess a five-letter word, with feedback given for each guess in the form of colored tiles indicating when letters match or occupy the correct position. After every guess, each letter is marked as either green, yellow or gray: green indicates that letter is correct and in the correct position, yellow means it is in the answer but not in the right position, while gray indicates it is not in the answer at all. Multiple instances of the same letter in a guess, such as the "o"s in "robot", will be colored green or yellow only if the letter also appears multiple times in the answer; otherwise, excess repeating letters will be colored gray (Wikipedia contributors, 2022b).

Let us define a "Wordle set" as the set of five-letter words in `wordle.txt` that match a given configuration of green, yellow, and gray letters. Complete function `exercise9(green,yellow,gray)` that returns the cardinality (i.e. the size) of the corresponding Wordle set. In particular, `green` is a dictionary that specifies the letters whose positions are known. The keys are positions (i.e., 0,1,2,3,4) and their associated values are their corresponding letters. Only positions with known letters are included in the dictionary. `yellow` is a dictionary specifying the letters that are in the answer but their position is not known. The keys are letters and their values are sets of known wrong positions (i.e., 0,1,2,3,4). Finally, `gray` is a set of letters that are known to not be in the answer. Note that a letter cannot be in `gray` and, at the same time, in `green` or in `yellow`. However, a letter could be both in `green` and in `yellow` and the clues could refer to the same letter.

Example: Given `green = {1:'i',3:'c'}`, `yellow = {'e':{3}}`, and `gray = {'r','a','s','d','f'}`, `exercise9(green,yellow,gray)` returns 5, as the Wordle set is comprised of 'wince', 'mince', 'niece', 'piece', and 'yince'.

Exercise 10: One Step of Wordle

This exercise builds upon the previous one and, therefore, makes use of the provided `wordle.txt` file that contains a list of 5-letter words used in the game Wordle (Disclaimer: This list is taken directly from the Wordle game and therefore may contain some words which some people may find offensive. Reader discretion is advised).

Given the `green`, `yellow` and `gray` letters, `exercise10(green,yellow,gray)` returns the set of 'best words,' according to the criterion of choosing the

[2]Online: `https://www.nytimes.com/games/wordle/index.html`. Last access October 28, 2022.

word(s) that provide the most information under every possible scenario, explained in the following.

Given `green`, `yellow`, and `gray`, compute their associated Wordle set S. For every word v in S, calculate its score as follows.

1. Consider every other word w in S and assume that it is the correct one. Update the configuration of the green, yellow, and gray letters with the information provided by w using the following rules:

   - If $v_i = w_i$, add the item $i : v_i$ to `green`.
   - If $v_i \neq w_i$ and v_i is in w, then add to `yellow` an item i to the set corresponding to key v_i.
   - If $v_i \neq w_i$ and v_i is not in w, then add an item v_i to `gray`.

   v_i and w_i refer to the letters in position i in the words v and w, respectively.

2. Use the new configuration to calculate the hypothetical Wordle set $T(v, w)$ and its cardinality $|T(v, w)|$.

3. The score of v is the sum over w of all the cardinalities $|T(v, w)|$.

The set of best words $v^\star$ is the set of words having the lowest score. Note that this set might have only one element.

Examples: Given the same `green`, `yellow` and `gray` as in the Exercise 9's example, the set of best words is: `{'wince', 'yince', 'mince'}`. In fact, the scores obtained are: `'niece'` 10; `'yince'`, 6; `'piece'`, 10; `'wince'`, 6; `'mince'`, 6.

Learning Outcomes Assessed

LO1: Use high-level programming languages to complete programming tasks.

LO2: Demonstrate familiarity with programming concepts, simple data-structures and algorithms

Criteria for assessment

Credit will be awarded against the following criteria.

Each exercise can be awarded a maximum of 10 marks. Therefore, perfectly solving five exercises will give you 50 marks (pass), and perfectly solving ten exercises will give you 100 marks.

Each exercise is marked for both function (8 marks max) and style/quality (2 marks max). Exercises that are not a real attempt at solving the problem presented will receive zero marks.

Functionality [8 marks/exercise max] The functional part of the submission is automatically marked by scripts that run the completed function against a set of test cases. You will be provided with two test cases per exercise. During marking, each exercise will be tested against four test cases, including the two provided. A test should take less than three seconds to complete, otherwise, it is considered failed. For each language (Python and JavaScript) and exercise, passing one test will award you 1 mark. Therefore, the maximum functionality mark of 8 will be given only if all the tests are passed in both languages.

Code quality and style [2 marks/exercise max] Each version of the exercise (i.e., Python and JavaScript) is assessed independently, according to the following criteria:

High quality and style (50-100%, 0.5-1 marks per exercise)	Low quality and style (0-50%, 0-0.5 marks per exercise)
Code is elegant	Code is messy or overly verbose
Code has no redundancies	Code has multiple redundancies and repetitions
Code is well commented	Code is lacking in meaningful comments
Code is perfectly modular (i.e., appropriate functions and/or classes defined)	Code is disorganised
Code makes smart use of built-in language features and classes.	Code does not make use of language features

Therefore, an exercise solved in only one language could achieve at most one of the style/quality marks. To obtain both marks, the exercise must be solved in both languages. Also, only exercises that pass at least two tests are evaluated for style and quality.

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on the return date via Learning Central and/or email.

The feedback from this assignment will be useful for your second programming assignment, and will also be relevant for any future programming tasks.

Submission Instructions

All coursework should be submitted via upload to Learning Central.

Description	Type	Name
Python Code	1 `.py` file	`[Student number].py`
JavaScript Code	1 `.js` file	`[Student number].js`

Any code submitted will be run on a system equivalent to the University provided Windows laptop, and must be submitted as stipulated in the instructions above. The code should run without any changes being required to the submitted code, including editing of filenames.

Any deviation from the submission instructions above (including the number and types of files submitted) may result in a reduction in marks for that assessment or question part of 10%.

Staff reserve the right to invite students to a meeting to discuss coursework submissions.

Support for assessment

Questions about the assessment can be asked on https://stackoverflow.com/c/comsc/ and tagged with '[CMT120]', during Tutorial lectures in Weeks 5-11, and in the Drop-in Practical Support sessions. It is important to notice that only general guidance will be provided. The tasks of solving the exercises, defining the logic and debugging the code (among others) are the key focus of the assessmentt and therefore a responsibility of the students.

References

Wikipedia contributors. (2022a). *Finite-state machine — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/wiki/Finite-state_machine` ([Online; accessed October 28, 2022])

Wikipedia contributors. (2022b). *Wordle — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/wiki/Wordle` ([Online; accessed October 28, 2022])