# Cardiff School of Computer Science and Informatics

## Coursework Assessment Pro-forma

**Module Code:** CMT120

**Module Title:** Fundamentals of Programming

**Lecturers:** Federico Liberatore

**Assessment Title:** Programming Challenges

**Date Set:** 23rd October 2023 **Submission date and Time:** 14th December 2023 at 9:30AM **Return Date:** 15th January 2024

**If you have been granted an extension for Extenuating Cir- cumstances, then the submission deadline and return date will be later than that stated above. You will be advised of your revised submission deadline when/if your extension is approved.**

**If you defer an Autumn or Spring semester assessment, you may fail a module and have to resit the failed or deferred compo- nents.**

**If you have been granted a deferral for Extenuating Circum- stances, then you will be assessed in the next scheduled assessment period in which assessment for this module is carried out.**

**If you have deferred an Autumn or Spring assessment and are eligible to undertake summer resits, you will complete the deferred assessment in the summer resit period.**

**If you are required to repeat the year or have deferred an as- sessment in the resit period, you will complete the assessment in the next academic year.**

**As a general rule, students can only resit 60 failed credits in the summer assessment period (see section 3.4 of the** [**academic**](https://intranet.cardiff.ac.uk/students/study/your-rights-and-responsibilities/academic-regulations)[**regulations).**](https://intranet.cardiff.ac.uk/students/study/your-rights-and-responsibilities/academic-regulations) **Those with more than 60 failed credits (and no more than 100 credits for undergraduate programmes and 105 credits for postgraduate programmes) will be required to repeat the year. There are some exceptions to this rule and they are applied on a case-by-case basis at the exam board.**

**If you are an MSc student, please note that deferring assess- ments may impact the start date of your dissertation. This is**

**because you must pass all taught modules before you can begin your dissertation. If you are an overseas student, any delay may have consequences for your visa, especially if it is your intention to apply for a post study work visa after the completion of your programme.**

**NOTE: The summer resit period is short and support from staff will be minimal. Therefore, if the number of assessments is high, this can be an intense period of work.**

This assignment is worth 40% of the total marks available for this mod- ule. 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.](https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/extenuating-circumstances) Only students with ap- proved 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/](https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/extenuating-circumstances) [extenuating-circumstances](https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/extenuating-circumstances)

[https://intranet.cardiff.ac.uk/students/study/your-rights-and-responsibilities/](https://intranet.cardiff.ac.uk/students/study/your-rights-and-responsibilities/academic-regulations) [academic-regulations](https://intranet.cardiff.ac.uk/students/study/your-rights-and-responsibilities/academic-regulations)

By submitting this assignment you are accepting the terms of the fol- lowing 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 declare that I have not made unauthorised use of AI chatbots or tools to complete this work, except where permitted. 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.*

# Assignment

To complete this coursework, you must complete a set of programming chal- lenges 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 per- fectly 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 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. Note that there is a time limit of 3 seconds to each test. If your code does not provide a correct answer within the time limit the test is failed.

### 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 func- tions’ 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 mod- ule 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 within the time limit. Further marks will be awarded for solution style and quality. The mark scheme is described in further detail later.

## Exercise 1: Smallest Fraction Terms

Complete the function ‘exercise1’ that takes two positive integers as its parameters. The parameters represent the numerator and denominator of a fraction. The function reduces the fraction to lowest terms and then returns both the numerator and the denominator of the reduced fraction as integers. **Python:** The numerator and denominator should be returned as a tuple.

**JavaScript:** The numerator and denominator should be returned as elements 0 and 1 of a 2-item array.

### Examples:

* exercise1(12,15) returns (4, 5) in Python and [4,5] in JavaScript.
* exercise1(8,4) returns (2,1) in Python and [2,1] in JavaScript.

## Exercise 2: Magical Dates

A magic date is a date where the day multiplied by the month is equal to the last two digits of the year.

For example 10/6/1960 is a magic date because 6 times 10 is 60, which is equal to the last two digits of 1960.

Complete the function ‘exercise2’ that takes three positive integers as its parameters, day, month, and year, respectively, and returns True if the date is a magic date, or False otherwise.

### Examples:

* exercise2(10,6,1960) returns True in Python and true in JavaScript.
* exercise2(10,6,1970) returns False in Python and false in JavaScript.

### Exercise 3: Find All Sublists

A sublist is a list that constitutes a portion of a larger list. A sublist can consist of a single element, multiple elements, or no elements at all.

**Example:** [1], [2], [3] and [4] are all sublists of [1, 2, 3, 4]. The

list [2, 3] is also a sublist of [1, 2, 3, 4], but [2, 4] is not a sublist of

[1, 2, 3, 4] because the elements 2 and 4 are not adjacent in the longer list. The empty list is a sublist of any list. As a result, [] is a sublist of [1, 2, 3, 4]. A list is a sublist of itself, meaning that [1, 2, 3, 4] is

also a sublist of [1, 2, 3, 4].

Using the above definition of a sublist, complete the function ‘exercise3’ that takes a list as its only parameter, and returns a list containing every possible sublist of the input list.

### Examples:

* exercise3([1,2,3,4]) returns

[[],[1],[2],[3],[4],[1,2],[2,3],[3,4],[1,2,3],[2,3,4],[1,2,3,4]].

* exercise3([’a’,2,(0,"zero")]) returns

[[],[’a’],[2],[(0,"zero")],[’a’, 2],[2, (0,"zero")],[’a’, 2, (0,"zero")]].

The order of the elements of the list returned is not important. However, the order of the elements inside of each list should reflect the order in the original list. For example, [’a’, 2] is correct, while [2, ’a’] is not.

### Exercise 4: English to Pig Latin Translator

Pig Latin is a language game or argot in which English words are altered, usually by adding a fabricated suffix or by moving the onset or initial con- sonant or consonant cluster of a word to the end of the word and adding a vocalic syllable to create such a suffix (Wikipedia).

The following rules are used to translate English into Pig Latin:

* If the word begins with a consonant (including ‘y’), then all letters at the beginning of the word, up to the first vowel (excluding ‘y’), are removed and then added to the end of the word, followed by ‘ay’. **Example:** ‘computer’ becomes ‘omputercay’ and ‘think’ becomes ‘ink- thay’.
* If the word begins with a vowel (not including ‘y’), then ‘way’ is added to the end of the word.

**Example:** ‘algorithm’ becomes ‘algorithmway’ and ‘office’ becomes ‘officeway’.

Complete the function ‘exercise4’ that takes a string as the only param- eter and return a string representing its Pig Latin translation.

### Examples:

* exercise4(‘algorithm’) returns ‘algorithmway’.
* exercise4(‘computer’) returns ‘omputercay’.

The function should correctly handle uppercase letters and punctuation marks such as commas, periods, question marks and exclamation marks. You can assume that only the first letter can be uppercase and that punc- tuation marks can only be at the end of the word.

**Example:** if an English word begins with an uppercase letter, then its Pig Latin representation should also begin with an uppercase letter and the uppercase letter moved to the end of the word should be changed to lowercase. For example, ‘Computer’ should become ‘Omputercay’. If a word ends in a punctuation mark, then the punctuation mark should remain at the end of the word after the transformation has been performed. For example, ‘Science!’ should become ‘Iencescay!’.

### Exercise 5: Morse Code Encoder

Morse code is a method used in telecommunication to encode text characters as standardized sequences of two different signal durations, called dots and

dashes (or dits and dahs). Morse code is named after Samuel Morse, an inventor of the telegraph (Wikipedia).

Complete the function ‘exercise5’ that takes a string of letters and num- bers as the only parameter and returns a string with its Morse code repre- sentation.

Use a period ‘.’ to represent a dot, and a minus sign ‘-’ to represent a dash. The mapping from letters and numbers to dashes and dots is illus- trated in the following figure.

A U

B V

C W

D X

E Y

F Z

G H I J

K 1

L 2

M 3

N 4

O 5

P 6

Q 7

R 8

S 9

T 0

The function should leave a single space between each sequence of dashes and dots and it should ignore any characters that are not letters or numbers.

### Example:

* exercise5(‘Hello, World!’) returns ‘.... . .-.. .-.. --- .-- --- .-. .-. ’.
* exercise4(‘Dracarys!!!!’) returns ‘-.. .-. .- -.-. .- .-. -.-- ’.

*Hint:* use dictionaries to represent the Morse code table.

### Exercise 6: Spelling Out Numbers

Complete the function ‘exercise6’ that takes an integer between 0 and 999 as its only parameter, and returns a string containing the English words for that number.

* exercise6(21) returns ‘twenty-one’.
* exercise6(191) returns ‘a hundred and ninety-one’.

*Hint:* Use one or more dictionaries to implement your solution rather than large if/elif/else constructs.

### Exercise 7: No Functions without Comments

Complete the function ‘exercise7’ that takes a string containing a filename as its only parameter. The filename should be a source file. The function reads the file, identifies functions that are not immediately preceded by a comment and returns a list of their name (represented as strings).

**Python:** For the purposes of this exercise, assume that any line that begins with ‘def’, followed by a space, is the beginning of a function defini- tion. Also, assume that, when the function has a comment, the comment character, ‘#’, will be the first character on the previous line.

**JavaScript:** For the purposes of this exercise, assume that any line that begins with ‘function’, followed by a space, is the beginning of a function definition. Also, assume that, when the function has a comment, the single- line comment symbol, ‘//’, will be the first character on the previous line.

### Exercise 8: Justify any Text

Complete the function ‘exercise8’ taking two parameters, a filename, and a maximum length (i.e., a strictly positive integer number). The function opens the file, reads every line, and returns a list of strings where every string represents a line that is filled as much as possible without exceeding the given maximum length.

**Example:** consider a file containing the following lines from ‘Alice’s Adventures in Wonderland’.

Alice was

beginning to get very tired of sitting by her sister

on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had

no

pictures or conversations in it,"and what is the use of a book," thought Alice, "without pictures or conversations?"

The corresponding output for length = 50 would be:

[’Alice was beginning to get very tired of sitting’, ’by her sister on the bank, and of having nothing’, ’to do: once or twice she had peeped into the book’,

’her sister was reading, but it had no pictures or’, ’conversations in it, "and what is the use of a’, ’book," thought Alice, "without pictures or’, ’conversations?"’]

You do not need to deal with multiple paragraphs of text. That is, you can group together words from different paragraphs. Finally, you can assume that no word is longer than the maximum length.

### Exercise 9: Knight’s Challenge

Complete the function ‘exercise9’ that takes three parameters: an initial position, a final position, and a number of moves. The function returns True if a knight on an empty chessboard can get to the final position from the starting position in at most the given number of moves; otherwise, the function returns False. Useful facts:

* A chessboard is an 8x8 square board.
* Each cell of the chessboard is identified by its coordinates: a letter from ‘a’ to ‘h’ that identifies the column, and a number from 1 to 8 that identifies the row. The positions are provided to the functions using this format.
* The chessboard contains only the knight, which is located at the spec- ified initial position.
* A knight may move two squares vertically and one square horizontally, or two squares horizontally and one square vertically.

### Examples:

* exercise9(’a1’, ’c5’, 2) returns True.
* exercise9(’c6’, ’h1’, 1) returns False.

*Hint:* You may want to use recursion.

### Exercise 10: War of Species

Complete the function ‘exercise10’ that takes as input a list of strings of the same length, representing a rectangle grid. The characters in the list represent the cells of the grid. The cells can take three possible values:

* ’X’, representing an individual of the species X.
* ’O’, representing an individual of the species O.
* ’.’, representing an empty cell.

### Example:

["X.......", "........", ". O"]

is a 3x8 grid with an ’X’ in position (1,1) and a ’O’ in position (3,8).

The configuration provided in the list represents the current state of an environment with two competing species. Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent. The function should return a list of strings representing the next state of the environment according to the following rules:

* An empty cell becomes non-empty if it is surrounded by at least two individuals of the same species. In particular, it becomes an individual of the most frequent species in its neighbourhood. In case of a draw between the species, the cell remains empty.
* A non-empty cell becomes empty if it is surrounded by more than six non-empty cells, regardless of their species.
* A non-empty cell becomes empty if it is surrounded by less than three members of its species.
* A non-empty cell becomes empty if it is surrounded by more members of the opposite species than members of its species.
* In any other circumstances, a cell does not change its value.

You can assume that the format of the input list of strings is correct.

**Example:** The initial environment

["XX......", "XX....O.", ". OOO"]

becomes

["XXX.....", "XXX..OOO", "XX. O."]

## 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, per- fectly 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 sub- mission 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 ex- ercise. 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 has no redundancies Code is well commented  Code is perfectly modular (i.e., appropriate functions and/or classes defined)  Code makes smart use of built-in  language features and classes. | Code is messy or overly verbose  Code has multiple redundancies and repetitions  Code is lacking in meaningful comments  Code is disorganised  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 pro- gramming 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 |

Replace [Student number] with your Cardiff’s user name, which is typ- ically a letter **’c’** (or **’d’**) + your student number, e.g. *c1234567*.

Any code submitted will be run on a system equivalent to the laptops provided to the students, and must be submitted as stipulated in the in- structions 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 deduction of up to 10% from the overall mark.

Staff reserve the right to invite students to a meeting to discuss course- work 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 3-11, 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 assessment and therefore a responsibility of the students.

**COMSC: Guidance on Referencing in Code.**

Cardiff University has in place an unfair practice procedure that covers what will happen if a student acts in a way “that might obtain for him/herself or for another, an unpermitted advantage or a higher mark or grade than his/her abilities would otherwise secure”. The guidance in this document, designed to help students avoid this, should be read in conjunction with:

* The information on Unfair Practice available on the Student Intranet: https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/sitting-your- exams/cheating-and-unfair-practice
* The Cardiff University regulations on unfair practices: [https://intranet.cardiff.ac.uk/intranet/students/documents/exams-and-assessment/1.11-](https://intranet.cardiff.ac.uk/intranet/students/documents/exams-and-assessment/1.11-Unfair-Practice-Procedure.pdf) [Unfair-Practice-Procedure.pdf](https://intranet.cardiff.ac.uk/intranet/students/documents/exams-and-assessment/1.11-Unfair-Practice-Procedure.pdf)

**Why would you copy code in an assignment?**

A lot of large-scale software development involves the reuse of smaller chunks of code, which is then adapted to work in conjunction with other small pieces of code. Sometimes this reused code is within a library module and sometimes that code is found on a tutorial web site like MDN or a Q&A site like Stack Overflow.

Depending on the assignment you are set, you may feel that you should reuse code from some of these sites in order to: hasten the development of your application, experiment with unused technology, or to fix a problem that you are struggling with.

Whether or not you should reuse code will depend on the assignment set. You will not gain any marks for writing the copied code (because you didn’t) but you will gain marks for integrating it into your solution therefore you must consider the aim and the marking criteria of the assignment.

As a simple example to illustrate the point:

Your Java assignment is to write a java programme to perform a bubble sort on an array of numbers.

You copy (and reference) a code snippet from Stack Overflow that performs a bubble sort on an array of numbers carefully putting the algorithm into the method signature required.

This has missed the point of the exercise, you gain no marks for the production of the algorithm and the integration is very minimal so will probably get no marks for that.

However, if:

Your Commercial Applications assignment is to create a website that will collect student grade data, store it, and be able to export that formatted data in a JSON file sorted by grade.

You copy (and reference) a code snippet from Stack Overflow that performs a bubble sort on an array of numbers carefully putting the algorithm into a method signature, then integrate this method into your project.

You will not get the marks for coding a bubble sort, but you will get the marks for the integration of that code into a bigger project. You will get the marks for deciding to use a bubble sort (although, probably, more marks would have been given for a different approach). In all, unless the assignment specified otherwise, it is likely you would see little difference in your mark if you had written it yourself.

**How to Reference Code in Your Code**

Just as when you include other people’s ideas and words in your writing you must attribute them to the author, when you include other people’s code within your code you must do the same; otherwise, you are pretending that this is your own work.

If your lecturer does not explicitly define how they wish you to cite any copied code there are some general examples given below.

It does not take much to reference code in your code. If you have copied a chunk of code then you can reference this in a comment, close to where the copied code has been included. You should include as a minimum some general information about the code copied, the URL where you copied the code from, and the date accessed e.g.

// Code to perform bubble sort

// taken from Stack Overflow post by A.C Oder 20-6-2004

// accessed 4-9-2018

// [https://stackoverflow.com/questions/. /bubblesort-implementation](https://stackoverflow.com/questions/........./bubblesort-implementation)

[[ CODE here ]]

// end of referenced code.

OR

You could include this information in the method or function’s documentation comment, e.g.

/\*\*

* Returns a sorted array
* This code was taken from Stack Overflow post by A.C Oder 20-6-2004
* accessed 4-9-2018
* [https://stackoverflow.com/questions/. /bubblesort-implementation](https://stackoverflow.com/questions/........./bubblesort-implementation)
* [@param](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#%40param) input unsorted array
* [@return](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#%40return) sorted array

\*/

You may have had to adapt a lot of the copied code to make it fit with what you needed. In which case you could state that the code has been adapted and highlight the lines / chunks you change so that the marker can credit you for that work.

/\*\*

* Returns a sorted array
* This code was adapted from Stack Overflow post by A.C Oder 20-6-2004
* accessed 4-9-2018
* [https://stackoverflow.com/questions/. /bubblesort-implementation](https://stackoverflow.com/questions/........./bubblesort-implementation)
* Added code to allow for sorting high to low or low to high
* see further comments for adaptions
* [@param](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#%40param) input unsorted array
* [@param](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#%40param) true high to low, false low to high
* [@return](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#%40return) sorted array

\*/

Then in the code you should highlight the parts that you wish to gain credit for adapting via additional comments.

**Further resources:**

A guide from MIT on referencing in code: <http://integrity.mit.edu/handbook/writing-code>