

**SCCT**

**aCADEMIC YEAR 2024/25**

Assessment Brief

# Submission and feedback dates

**Submission deadline:** **Before 14:00 on 3rd April 2025**

Is eligible for 48-hour late submission window

**Marks and Feedback due on:** 26th May 2025

N.B. all times are 24-hour clock, current local time (at time of submission) in the UK

# Submission details

**Module title and code**: UFCFYR-15-2 Advanced Algorithms

**Assessment type**: Coursework

**Assessment title:** Individual Coursework with the video demo

**Assessment weighting:** 50% of total module mark

**Module learning outcomes assessed by this task:**

1. Formulate problems as abstract models which can be solved by generic algorithms and mathematical methods.
2. Critically evaluate, the effectiveness of the design, efficiency of the applications of algorithms for processing data on a wide range of problems.
3. Execute and implement algorithms in a programming language.

Table of Contents

[**aCADEMIC YEAR 2024/25** 1](#_Toc189686387)

[Submission and feedback dates 1](#_Toc189686388)

[Submission details 1](#_Toc189686389)

[**Section 1: What am I required to do on this assessment?** 2](#_Toc189686390)

[**Section 1.1 Task specification** 2](#_Toc189686391)

[**Section 1.2 Deliverables** 5](#_Toc189686392)

[**Section 2: Where should I start?** 5](#_Toc189686393)

[**Section 3: What do I need to do to pass? (Marking Criteria)** 6](#_Toc189686394)

[**Section 4: How do I achieve high marks in this assessment?** 6](#_Toc189686395)

[**Section 5: How does the learning and teaching relate to the assessment?** 7](#_Toc189686396)

[**Section 6: What additional resources may help me complete this assessment?** 7](#_Toc189686397)

[**Section 7: What do I do if I am concerned about completing this assessment?** 7](#_Toc189686398)

[**Section 8: Marks and Feedback** 7](#_Toc189686399)

## **Section 1: What am I required to do on this assessment?**

Broadly speaking, the assignment requires you to select and implement appropriate data structures to implement designed algorithms to solve the specific problems and satisfy system requirements such as high performance and reasonable memory space.

### **Section 1.1 Task specification**

**Task 1: (90%)**

**Task 1.1: (20%) Degree Calculation**

We are going to develop a final degree calculation system for UWE. The main functionality of the system is to import required data and automatically calculate the degree marks and bands. The calculation is based on the UWE Academic Regulations 23/24 as follows:

*The University awards its degrees as follows:*

*• First Class: 70% or higher.*

*• Second Class (Upper Division) (2.1): 60% to less than 70%.*

*• Second Class (Lower Division) (2.2): 50% to less than 60%.*

*• Third Class: 40% to less than 50%.*

*• A student who achieves less than 40% will fail.*

*A student must have passed all modules from year one, two and three to get the degree in accordance with the Regulations.*

*The following actions are taken to determine the degree classification:*

*1. A student’s weighted average mark for Level 5 (year two) is calculated by selecting the best marks across 100 credits at Level 5, multiplying the credit value of each module by the percentage mark achieved for that module, and the sum of those results is then divided by the total number of credits. Pass/fail modules are excluded from the calculation.*

*2. A student’s weighted average mark for Level 6 (year three) is calculated by multiplying the credit value of each module by the percentage mark achieved for that module, and the sum of those results is then divided by the total number of credits. Pass/fail modules are excluded from the calculation.*

*3. The final aggregate mark used for degree classification will comprise the weighted average % mark of Levels 6 and 5 weighted in the ratio 3:1. This means that the final aggregate % mark is calculated by multiplying the weighted average % mark of Level 6 by 3 and adding it to the weighted average % mark of Level 5 and dividing the result by 4.*

We will use Computer Science students’ results as the testing. Therefore, two csv files are provided; one includes the module names and codes, the other includes 300 students’ module marks for three years.

Your system should be able to import these two files or the similar data from other courses, and the calculated degree marks and related information should be saved in a file.

**Task 1.2: (25%) Password Generator**

Suppose you are a skilled hacker and are going to crack a password by a brute force attack (an attempt to try every possible password). Fortunately, you have obtained the partial knowledge about the composition of the passwords, which can reduce the number of the potential passwords significantly.

A valid password consists of the elements from a set of capital letters {A, B, C, D, E}, a set of lower-case letters {a, b, c, d, e}, a set of digits {1, 2, 3, 4, 5} and a set of special symbols {$, &, %}. However, a collection of rules for composing a valid password are also gained as follows:

* it must include at least one element from each category;
* it must start with letters (capital or lower-case);
* it must not include more than two capital letters;
* it must not include more than two special symbols.

You task is to write a Python program, which takes a given number as the password length, and output all valid passwords (including indices) with the given length. For example, if the given length is 4, your program may return the follow result:

1 Aa1$

2 Ba1$

3 Ca1$

…

These sets (letters, digits, and special symbols) can be hard-coded or read in if you prefer, but the given length should be read in from the console for the easy testing.

The generated passwords should be saved in a file.

**Task 1.3: (20%) Train Ticket Search**

The national rail company invites you to design a train ticket search system. Simply speaking, users can input the names of the departure and destination stations, and the system returns the cheapest train ticket or the fastest one by the user selections. The route information, including all the stations it passes through, should be provided.

The company provides a csv file (railway\_network.csv) which includes the direct connections of the adjacent stations and their weights (costs and journey time). For example, the tuple (Bristol Temple Meads, Bristol Parkway, 10, 15) denotes that the cost between the two stations is 10 for the either way, and the journey time is 15 minutes.

This data is based on the existing national rail network that is provided as a PDF map. You can pick up any two station names from the map to test your program.

**Since there are too many stations in London, we have merged all these stations into ONE station called ‘London’ in the provided csv file.** That is, London is a hub to connect all lines from different directions.

**Task 1.4: (25%) Parallel Programming**

A company needs your help to develop a new program with parallel programming to significantly increase the efficiency of its legacy program. The provided program uses the ‘face\_recognition’ library to search the images from a set of unknown images (which can be found in the imageset folder) with a known face. However, this legacy program can support the serial search only.

Since the company has more CPU resources, it requires you to develop a parallelised program to speed up the searching process.

The serial program (task1\_4\_serial.py) is provided, and you should modify it into a program with TRUE parallelism. Efficiency and memory space are the major concerns for the marking criteria.

**Task 2 (10%)**

You should produce one mini report on algorithm design and one short video to demonstrate your programs.

**Task 2.1 (5%)**

Explain your design for all the tasks (1.1 – 1.4) in Task 1 such as the choice of data structure and the logic of your algorithms. You can use pseudocode or workflow diagrams to illustrate your algorithms if necessary, and a pure and clear narrative is also accepted as long as you can make readers understand your design with ease.

You need to focus on the key algorithms only and don’t need to explain any well-known algorithm or secondary procedures. For example, if you use merge sort or binary search, you just mention the algorithm names as we all know what these algorithms are. If you import data from a csv file, you don’t need to explain how the program reads it line by line.

You also need to justify your design choices in terms of effectiveness, efficiency or memory spaces of the system that you have produced. Please check the marking criteria carefully for each task.

You should mention any unoriginal work. For example, if you use any Python library, you should address it in this task. If you use any significant code from open-source projects, you should remark it. Please feel free to use the code from the lab exercises.

The mini report should be less than 800 words except for the pseudocode and diagrams.

**Task 2.2 (5%)**

Produce a short video (less than 8 minutes) to demonstrate your program running and implement each task in order. In the video, you can also emphasize any outstanding design in your program to the markers. Please give verbal narration, if possible, to explain your operations.

### **Section 1.2 Deliverables**

One folder in zip format (only) must be uploaded via the relevant link on the module’s space on Blackboard. The zipped file should be names as ‘AACoursework\_Your Name\_Student Number.zip’.

The link will be available two weeks before the due date and will be communicated to students via an email announcement.

You can produce an independent Python code for each sub-task. So, for this format, GUI or a simple menu is unnecessary.

The submission must contain:

* Python programs (all the python file names must be ended with **.py** and other python program patterns are not accepted) for Task 1.1 to Task 1.4.
* If non-standard Python libraries are used in your programs, the instruction like readme.txt should be included about how to install them.
* A word / PDF file for Task 2.1.
* A short video for Task 2.2 demonstrating your software running and fulfilling each of the tasks in the correct order.

All program files are saved in Python format (above 3.0). Any other version or other programming languages will not be accepted and will not be marked, which results in a zero mark for this coursework.

You are allowed to use any Python data structure or internal or external libraries. However, we do encourage you to write your own algorithm and use less libraries. The marking criteria is based on your contribution (more own code and more contribution) with the consideration of efficiency. Please check Section 3 for the marking criteria.

## **Section 2: Where should I start?**

First, you need to read the task description thoroughly, understand the given output of the task in terms of the related input and work out the correct output for any input.

Second, you should consider the algorithm for solving the task with the pseudocode. Meanwhile, you should think about appropriate data structures and efficient algorithms with your best knowledge. It is recommended that you can do your own research by means of peer discussion and Internet resources (Google, Code Forum, YouTube, ChatGPT and so on). Please modify the on-line code to adapt it to your own program, and direct copy and paste is considered a potential plagiarism.

Third, our teaching materials including lab exercises cover a wide range of data structures and algorithms, and provides pros and cons of them, and scenarios for usage. Please feel free to use it directly.

Finally, you should test your programs completely using your own data or provided testing data.

## **Section 3: What do I need to do to pass? (Marking Criteria)**

To pass the coursework, you must get 40% or above. Please check the marking criteria at the end of the coursework specification.

Note that the marking criteria for each mark band is incremental. For example, the 70+ band requires the 60+ band’s criteria automatically.

## **Section 4: How do I achieve high marks in this assessment?**

Achieving high marks in an assessment requires a combination of effective study strategies, time management, and a deep understanding of the subject matter. Here are some tips to help you excel in your assessment:

1. **Understand the Assessment Criteria**: carefully review the assessment guidelines and criteria provided by your instructor or syllabus. Understand what is expected in terms of content, format, and grading criteria.
2. **Create a Study Plan**: develop a structured study plan that outlines what topics you need to cover, how much time to allocate to each, and when to start studying. Prioritize difficult or unfamiliar topics.
3. **Take Effective Notes**: when attending lectures or studying from textbooks, take organized and concise notes.
4. **Practice Active Learning**: engage actively with the material by asking questions, making connections, and summarizing key concepts in your own words. This helps deepen your understanding.
5. **Seek Clarification**: if you have questions or encounter challenging topics, don't hesitate to seek clarification from your instructor, classmates, or online resources. Understanding difficult concepts is crucial.
6. **Manage Your Time**: allocate sufficient time for studying and avoid last-minute cramming. Regular, spaced study sessions are more effective than long, stressful cramming sessions.
7. **Study Groups**: discussing and teaching concepts to others can enhance your understanding.

## **Section 5: How does the learning and teaching relate to the assessment?**

The lecture slides, lab exercises and related tutorials from Week 1 to Week 9 are sufficient for finishing the coursework well.

## **Section 6: What additional resources may help me complete this assessment?**

There are a number of internet resources which can help you finish the coursework well. However, please do use it correctly to avoid the plagiarism allegation. Please contact your lab tutor if you have any concern about the on-line resources.

Please take advantage of the lab sessions and you can get prompt and in-person feedback from the lab tutor if you present some of your work.

You can also make the appointment with the module leader to discuss any issues about your coursework for a further clarification.

The frequently asked questions will be published on BB if the module leader thinks it’s necessary to announce to all.

For the general study skill, you can refer to UWE library page <https://www.uwe.ac.uk/study/study-support/study-skills>

## **Section 7: What do I do if I am concerned about completing this assessment?**

UWE Bristol offer a range of Assessment Support Options that you can explore through [this link](https://www.uwe.ac.uk/study/academic-information/personal-circumstances), and both [Academic Support](https://www.uwe.ac.uk/study/study-support/student-support-advisers) and [Wellbeing Support](https://www.uwe.ac.uk/life/health-and-wellbeing/get-wellbeing-support) are available.

For further information, please see the [Academic Survival Guide](https://www.uwe.ac.uk/study/academic-information/academic-survival-guide).

## 

## **Section 8: Marks and Feedback**

Your assessment will be marked according to the marking criteria represented in Section 3. The formal feedback will be returned to the students with the mark within four weeks after the submission. Please check the exact date from the cover page.

You can also contact the markers or the module leader about the clarification of the feedback and the mark when they are published.

There are more general advices before the coursework submission.

1. In line with UWE Bristol’s [Assessment Content Limit Policy](https://www.uwe.ac.uk/about/structure-and-governance/policies) (formerly the Word Count Policy), word count includes all text, including (but not limited to): the main body of text (including headings), all citations (both in and out of brackets), text boxes, tables and graphs, figures and diagrams, quotes, lists.
2. UWE Bristol’s [UWE’s Assessment Offences Policy](https://www.uwe.ac.uk/study/academic-information/assessments/assessment-offences) requires that you submit work that is entirely your own and reflects your own learning, so it is important to:
   * Ensure you reference all sources used, using the [UWE Harvard](https://www.uwe.ac.uk/study/study-support/study-skills/referencing/uwe-bristol-harvard)/[OSCOLA](https://www.uwe.ac.uk/study/study-support/study-skills/referencing/oscola) system and the guidance available on [UWE’s Study Skills referencing pages](https://www.uwe.ac.uk/study/study-support/study-skills/referencing).
   * Avoid copying and pasting any work into this assessment, including your own previous assessments, work from other students or internet sources
   * Develop your own style, arguments and wording, so avoid copying sources and changing individual words but keeping, essentially, the same sentences and/or structures from other sources
   * Never give your work to others who may copy it
   * If an individual assessment, develop your own work and preparation, and do not allow anyone to make amends on your work (including proof-readers, who may highlight issues but not edit the work) and

**When submitting your work, you will be required to confirm that the work is your own,** and text-matching software and other methods are routinely used to check submissions against other submissions to the university and internet sources. Details of what constitutes plagiarism and how to avoid it can be found on UWE’s Study Skills [pages about avoiding plagiarism](https://www.uwe.ac.uk/study/study-support/study-skills/reading-and-writing/plagiarism).

**Marking Criteria**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0-29 | 30-39 | 40-49 | 50-59 | 60-69 | 70-84 | 85-100 |
| **Task 1.1**  **20%** | The program does not run or does not deliver a complete solution. | The calculation formula is not well built, and so the degree calculation is not accurate. | Data is imported correctly. The output is accurate for most cases. | The program can always output correct results, but it’s not efficient. E.g., unnecessary ‘for loops’ or wasted variables and data structure. | The program can always output correct results efficiently. But the output is not well presented. E.g., results are not stored in a file or include limited information. | The program can always output correct results efficiently with beautiful presentation (mark details). The code is well commented and structured. The code can read in different data. | The program is very professional with easy maintenance or modification of calculation rules. |
| **Task 1.2**  **25%** | The program does not run or does not deliver a complete solution. | The program runs but cannot print the passwords with the given length or fail all the rules. | Passwords are generated by Python libraries but fails one or more rules. | Passwords are generated by Python libraries and satisfies all the rules. The algorithm is not efficient. | Python libraries are used to generate the passwords, but efficiently. Or develop own algorithm to generate passwords but relatively efficient. | Develop own algorithm to generate passwords efficiently. | Develop own algorithm to generate passwords with a superb efficiency. |
| **Task 1.3**  **20%** | The program does not run or does not deliver a complete solution. | The program runs and does deliver a complete solution. However, the output is incorrect at all. | Data are imported and stored well. The correct algorithm is used, but the output is not always correct. E.g., one of the two search options is incorrect. | Data are imported and stored well. The correct algorithm is used, and the output is correct if input names are correct, but the route is not provided. | The program is elegant and professional such as good comments, meaning variable names and useful functions. The output satisfies the requirement. | Extra features are provided to help the user experience such as convenient or fast input and even allow the modification of the raw data. | The design is beyond the requirement such as search by similarity of station names. |
| **Task 1.4**  **25%** | The program does not run or does not deliver a complete solution. | The program does run and try hard to provide parallel processing. However, it delivers a serial process with an incorrect output. | The program does run and try hard to provide parallel processing. However, it delivers a serial process with a correct output. | The program provides a true parallelism with a correct output. However, it’s not efficient such as a fixed process number and a high overhead. | The program provides a true parallelism with a correct output. The efficiency is decent such as evaluation of local resources and proper workload allocation. | The program provides a true parallelism with a correct output. The efficiency is high such as workload distribution is considered for each process | The design is beyond the requirement. E.g., distributed or message passing models are considered. |
| **Task 2.1**  **5%** | The justification of the design choices is none or minimal. | The justification of the design choices is minimal and up to 40% of the tasks. Clarification and discussion are shallow. | The justification and clarification of the design choices are reasonable and cover up to 60% of the tasks. | The justification and clarification of the design choices are reasonable and cover up to 80% of the tasks. | The justification and clarification of the design choices are reasonable and cover all the tasks. | The justification and clarification of the design choices are excellent and profound and cover all the tasks. | An excellent discussion, fully evidenced and very well supported by relevant references. |
| **Task 2.2**  **5%** | Video demo is none or minimal | Video demonstrates up to 40% of the tasks | Video demonstrates up to 60% of the tasks | Video demonstrates up to 80% of the tasks | Video demonstrates up to all the tasks | Clarification is clear and concise, and data structure and algorithms are explained, and main features are emphasised. | Professional video presentation |