 Computer Science and Creative Technologies

**Coursework or Assessment Specification**

## Module Details

|  |  |
| --- | --- |
| **Module Code** | UFCFYR-15-2 |
| **Module Title** | Advanced Algorithms |
| **Module Leader** | Dr Kun Wei |
| **Module Tutors** | Dr Rong Yang and Dr Antisthenis Tsompanas |
| **Year** | 2022-23 |
| **Component/Element Number** | B |
| **Total number of assessments for this module** | 1 |
| **Weighting** | 100% |
| **Element Description** | Mainsit Coursework |

## Dates

|  |  |
| --- | --- |
| **Date issued to students** | 22nd Feb 2023 |
| **Date to be returned to students** | 5th June 2023 (mark publication) |
| **Submission Date** | 9th May 2022 (14 May with the grace period) |
| **Submission Place** | Blackboard – Online |
| **Submission Time** | 14:00 |
| **Submission Notes** | Please submit all files in a zipped folder **clearly labelled “AACoursework\_Your Name\_Student Number.zip ”** |

## Feedback

|  |  |
| --- | --- |
| **Feedback provision will be** | Formative / Verbal, during practical sessions. Written on return of the marked work. |

# Contents

[Module Details 1](#_Toc95941794)

[Dates 1](#_Toc95941795)

[Feedback 1](#_Toc95941796)

[Contents 2](#_Toc95941797)

[Section 1: Overview of Assessment 3](#_Toc95941798)

[Section 2: Task Specification 3](#_Toc95941799)

[Section 3: Deliverables 6](#_Toc95941800)

[Section 4: Academic Offences 7](#_Toc95941801)

[Section 5: Marking Criteria 7](#_Toc95941802)

[Section 6: Feedback mechanisms 8](#_Toc95941803)

[Marking Criteria Table 9](#_Toc95941804)



# Section 1: Overview of Assessment

This assignment assesses the following module learning outcomes:

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.

The assignment is worth **50%** of the overall mark for the module.

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 usage.

# 

# Section 2: Task Specification

There are two tasks in total. Task 1 consists of four subtasks and Task 2 includes two mini reports and one video demonstration. The assessment weight of each task is given in the following specifications.

Note that the marking criteria are based on the efficiency and other compulsory requirements of the program rather than simply whether your program can generate the correct results or not. Please read the marking criteria carefully and thoroughly for each task before you start.

**Task 1: (80%)**

**Task 1.1 (20%) remove duplicated elements with a special rule**

You are requested to develop an algorithm to read in a large amount of numbers, remove duplicated numbers with a bespoke rule, and write it back to a new file. However, your algorithm **must** follow the special rule to determine which unique element will be reserved.

**Rule:**

The middle element in the sequence of the duplicated elements can be retained only. For example, the list like [1, 2, **1**, 3, 1] is processed and its output should be [2, **1**, 3] because there are three ‘1’ in the list, and the red one can be retained as it’s the middle one, and the first and third ones are removed. If the number of the duplicated elements is even, the right one over the middle is retained. For example, the output of [ 1, 2, 1, 3, **1**, 1] is [2, 3, **1**] because there are four ones, and the third (or red) one is the one just over the middle.

There are more examples as follows to help you to understand the rule. The red elements in the lists are the elements which should be retained.

[2, 1, 2, 3, 1] -> [2, 1, **2**, **3**, **1**] -> [2, 3, 1]

[3, 2, 1] -> [**3**, **2**, **1**] -> [3, 2, 1]

[1, 2, 3, 3, 2, 1] -> [1, 2, 3, **3**, **2**, **1**] -> [3, 2, 1]

[3, 2, 1, 1, 2, 3, 2, 1, 3, 2] -> [3, 2, 1, **1,** 2**, 3**, **2**, 1, 3, 2] -> [1, 3, 2]

You should design an efficient algorithm to implement this removing operation or pick-up operation with the consideration of fast execution and less memory usage. The higher mark is rewarded in terms of how efficient your program is and how much memory the program uses (less is better).

A testing file with massive numbers is provided for testing the speed of the execution.

**Task 1.2: (20%) data operations**

We are implementing some basic data operations such as search, insertion and deletion. However, efficiency or execution speed is the only marking criterion.

A file with massive integers is provided. You are required to develop a program to read in all these numbers from the file, sort it and store it in a proper data structure.

Another operating file is provided, which consists of a sequence of data operations. Each operation includes a pair of numbers where the first number denotes which operation and the second one is the value to be processed. For operations, 1 denotes search, 2 is insertion and 3 is deletion. For example, (1, 10) is to search whether 10 is included in the imported numbers, (2, 10) is to insert 10 as the ascending order, and (3, 10) means the deletion of 10 from the data structure. Note that for the deletion operation, if there are duplicated numbers, the deletion should remove all the duplicated numbers.

In short, this task includes the following requirements.

1. Import the integers from the provided file.
2. Sort these numbers in an ascending order.
3. Import the data operations from the provided file.
4. Implement these data operations in order.

The execution time should be calculated as the proof of the efficiency.

**Task 1.3: (20%) search with parallelism**

You need to design a frequency-counting program with parallelism programming. Again, you are given two files: one is the file with a long text, and another is a file including a set of names. Your task is to count the frequency of each name in the text. For example, the result is like ‘Harry Potter’ 102, ‘Ron Wesley’ 54 or so on.

The result which contains all the names and their frequencies should be saved in a new file. The program must be a parallelized program.

**Task 1.4: (20%) cheapest train tickets**

The national rail company requests you to design a train ticket search system. Simply speaking, users can input the names of the departure and destination, and the system returns the cheapest train ticket and all the station names of the route.

The company provides a csv file (railway\_network.csv) which includes the direct connections of the adjacent stations and their weights (costs). For example, the tuple (Bristol Temple Meads, Bristol Parkway, 10) denotes that the cost between the two stations is 10 for the either way. 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 merge all these stations into **ONE** station called ‘London’. That is, London is a hub to connect all lines from different directions.

The requirements of the task are as follows:

1. Import the csv file.
2. Input two station names for departure and destination.
3. Return the cheapest cost of the two stations and all the station names on the route.

**Task 2 (20%)**

You should produce two mini reports and one short video to demonstrate your programs.

**Task 2.1 (10%)**

Explain your design for 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 general discuss should be less than **500 words** except for the pseudocode and diagrams.

**Task 2.2 (5%)**

Discuss any issue relating to computational sustainability that system developers should consider. Please use Task 1.4 to discuss the sustainability issue for the national railway services. Please give the reference if you use any formal definitions. This discuss should be less than **500 words.**

**Task 2.3 (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 the verbal narration, if possible, to explain your operations.

Please produce a file that contains clear instructions about how to run your program or how to set up the environment. It’s very important when you use some Python libraries. You don’t need to give any instruction if your programs are easy to run.

# Section 3: 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 for Task 1.1 to Task 1.4
* Any non-standard Python library used in your programs should be included. Your programs must be self-contained. Note that markers won’t install non-standard Python libraries to test your system. The failure of the implementation may be marked 0. Also, you should include all input data or testing data in the folder.
* A word / PDF file for Task 2.1.
* A word /PDF file for Task 2.2
* A short video for Task 2.3 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 program 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 library. However, you do need to give a brief introduction what they are and why to use them. In fact, there are many powerful built-in functions in Python data structures, and you feel free to use them. Again, you must talk about it when these functions may affect the efficiency to a great extent. For example, a Python list has a built-in sort function. You can use it directly rather than write the sort algorithm yourself. But you do need to mention the efficiency of the built-in sort function. That is, you should understand well anything that you use. From another hand, we encourage you to write your own sort algorithm.**

# Section 4: Academic Offences

This is an individual coursework, and therefore you should individually submit your own original work. In fact, we encourage you to work as a group or cohort. You are free to discuss the solutions for the coursework, however, you should then split up and work alone to finish the work. **The identical work is an offence**, **and the similar work with effortless changes (such as the change of variables names) is considered an offence too.**

Please do not share your code with anyone. Even if you don’t share your code on purpose, you are still considered a collusion if the similar work is found. For example, your friend may request your work for a reference, and promise not to copy your code. However, very often they may take the risk when the deadline is approaching. For this case, all the involved students may get the same penalty.

You may use the open-source code and projects or examples and tutorials from YouTube. Actually, we encourage you to take advantage of these on-line sources. **However, copy and paste of the on-line code is considered an offence.** This is very similar to your English writing. You can use references and we do use a lot of references. However, you cannot simply copy and paste the articles, and you must change it into your own words and give the reference links if necessary. For your own code, if you just use a fragment of on-line or public code to make it accommodative for your own solution, this is totally fine. If you simply add few lines into others’ programs and then claim this is your own work, this is problematic. Please talk to your tutors if you are unsure about how to use the on-line code.

The teaching team is professional about programming and marking. You might be invited for an interview to demonstrate and explain your program if the markers and the module leader doubt the originality of your work.

# Section 5: Marking Criteria

The following table (please see next page) gives details of the marking criteria for this coursework. Marks will be awarded for clear rationale justifying design choices.

Clarification in the system design submitted for Task 2.1 can help the markers to map the full logic of the solution into the correct marking section.

Code must be well structured, appropriately commented, neat and efficient. Clear use of functions and reduced repetitions of blocks of code are expected.

The marking band is incremental. For example, the claim of 70-85 means your work has satisfied any precedent band.

**NOTE – No hard coded data will be allowed. Hard coded data in the submitted work will result in the work marked at zero.**

# Section 6: Feedback mechanisms

Formative / Verbal will be provided during practical sessions. Written feedback will be provided on blackboard when the mark is published.

# Marking Criteria Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 program runs and can input or read in some numbers, however the duplicated elements preserve in the output. | No duplicated elements in the output, but the correct order is not reserved. | No duplicated elements and the correct order in the output. However, the algorithm is inefficient. | The output preserves the correct order with a reasonably efficient algorithm. Also, it does not use memory wisely and efficiently. | Suitable Data structure is applied with an efficient algorithm and least memory usage. | The design is beyond the requirement. The program is elegant and professional. |
| **Task 1.2**  **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 not always correct for the testing operations. | The program can return the correct results for all the testing operations, but not efficient. | The program can return the correct results for all the testing operations with the reasonable efficiency and data structure | The program can return the correct results for all the testing operations with the high efficiency and suitable data structure. | The program can return the correct results for all the testing operations with the extremely high efficiency and most suitable data structure. | The design is beyond the requirement. The program is elegant and professional. |
| **Task 1.3**  **20%** | 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 incorrect output. | The program does run and try hard to provide parallel processing. However, it delivers a serial process with correct output. | The program provides a parallel processing. However, the number of the processes is fixed for any size of the input. | The program provides a parallel processing. However, the number of the processes can be set up before dealing with the input. | The program can analyse the input to dynamically decide the number of the processes. | The design is beyond the requirement. E.g., dynamically allocate workload |
| **Task 1.4**  **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. | Data are imported and stored well. The correct algorithm is used, but the cost is 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 repeated input and even allow the modification of the raw data. | The design is beyond the requirement. |
| **Task 2.1**  **10%** | 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%** | The report is none or minimal for computational sustainability | The report discusses few points for computational sustainability, but they are irrelevant. | The report discusses some points for computational sustainability, but with minimal clarity | The report discusses some points for computational sustainability with reasonable clarity | The report gives comprehensive discussion, however, it isn’t related to the recommended or certain task> | The report gives comprehensive discussion with the combination of the tasks. | An excellent discussion, fully evidenced and very well supported by relevant references. |
| **Task 2.3**  **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 |