COMP3506/7505 Project - Part A

Weighting: 15%

Due date: 13th September 2021 at 4:00 PM AEST

Task

This project will require you to implement algorithms efficiently in order to solve desired tasks. Part A of this project will consist of two questions. For each of these questions you will be required to:

1. Complete programming task(s) in reference to a given algorithm or desired efficiency.
2. Write a report explaining the efficiency of your implementation.

The specific details required for each of these two points are outlined in the questions below.

Submission Details

All assignment related documents and files will be submitted via Gradescope. The programming sections of your assignment will be marked by auto-grading software while the report section of your assignment will be marked by a tutor. Therefore, you will need to submit to two separate Gradescope submission portals. The name of these submission portals and the documents required to submit to them are listed below.

- **Autograder:** You must submit the programming part of your assignment to the autograder portal. You should only submit either the java or python code according to the language you have used. You *must not* submit your report pdf here. If you do so and fail to submit the project PDF to the Report Portal (see below), penalties will apply.
- **Report:** You must submit a PDF export of your report to the report portal. You must use the template provided for your report. If you do not use the template or you modify the template in anyway, with the exemption of adding your answers in the box provided, you will incur a penalty.

Programming Details

Java instructions

- You must use Java version 11 or higher. Additional language features introduced after version 11 may not be supported. It is recommended you use OpenJDK, for example, AdoptOpenJDK 11.
- Your solution will be automatically marked. No marks will be awarded for non-compiling submissions.
- You should NOT use any additional classes from the Java Collections Framework (e.g. `ArrayList`, `LinkedList`, `HashMap`, `HashSet`). If you wish to utilise similar functionality, you should implement the needed data structures yourself. If you are unsure about whether a certain class is allowed to be used, please contact teaching staff immediately.
- You should NOT use any additional third-party libraries. No marks will be awarded to submissions which import non-supported libraries.

Python instructions

- You must use Python version 3.7 or higher.
- Your solution will be automatically marked. No marks will be awarded for non-compiling submissions.
- You should NOT use Python built-in sort methods, such as `list.sort()` or `sorted()`. You should NOT use Python built-in list methods, such as `append, clear, count, copy, extend, index, insert, pop, remove, reverse, sort`. You should NOT use dictionary `dict` nor {}. If you wish to utilise similar functionality, you should implement the needed methods yourself.
- You should NOT use any additional libraries, such as `numpy`, `scipy`, and `collections`. No marks will be awarded to submissions which import non-supported libraries.
- Remove `__main__` method and `print` functions before submitting to the autograder.

Late Submissions and Extensions

Please consult the courses ECP for the policies and procedures regarding late submission and extensions. Note that course staff cannot process requests for extension or otherwise. All such applications must be made through the school in accordance with ECP.

Academic Misconduct

This assignment is an individual assignment. Posting questions or copying answers from the internet is considered cheating, as is sharing your answers with classmates. All your work (including code) will be analysed by sophisticated plagiarism detection software.

Students are reminded of the University's policy on student misconduct, including plagiarism. See the course profile and the School web page: http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism.

Part A

1 Departure Display (40 Marks)

1.1 Context

The outbound flights in the airport need to be displayed on an electronic board for passengers to see. Each flight is defined by the flight number and the departure time. The flight number usually consists of 3 letters and 4 digits (e.g. "ABS1234"). The time is given as a string "hours:minutes" in military 24h format (e.g. "18:25", which is equivalent to 6:25 PM). The outbound flights need to be sorted by the departure time where the earliest flights come first. If two planes have the same departure time, they must be sorted in alphabetic order of their plane numbers. For simplicity we limit our system to one day.

1.2 Programming (20 Marks)

Assuming all the planes are not sorted, implement an efficient sorting algorithm to order the flights to display on the board. In the file `Display.java/display.py`, you should implement the methods in the interface `DisplayRandomBase`. You are provided with the constructor, that reads the unsorted list of planes in CSV format and stores it in one-dimensional array of `Plane` objects.

In the file `Plane.java/plane.py`, you should also implement any necessary methods in the `PlaneBase` interface.

In reality, the order of most planes is predefined by the timetable, and only a few planes are out of place. Update your sorting algorithm to take advantage of this fact. In the file `Display.java/display.py`, you should implement the methods in the interface `DisplayPartiallySortedBase`. You are provided with the constructor, that reads two lists of planes - sorted and unsorted - in CSV format and stores them in two one-dimensional arrays of `Plane` objects.

Your marks for this programming task will be calculated by running a series of tests against your solution using the gradescope autograder. Each test will be assigned a certain number of marks according to the rigour of the test. The total sum of all the tests will be 20 marks. The results of *some* tests will be available instantly after submission while others will be hidden until the assignment marks are released.

1.3 Report (20 Marks)

Using the gradescope document template provided, you must complete a report which analyses your code and the algorithms you have implemented. The report will be broken up into the following three sections:

1. Provide a screenshot of the main sorting function for the partially sorted list (DisplayPartiallySorted).
2. Describe which sorting algorithm you have used for the unsorted list and state its time and space complexity with respect to the number of planes, **n**.
3. Describe the modifications you have made to take advantage of the list being partially sorted. State the complexity of your algorithm with respect to **n** and **k**, where **n** is the number of sorted planes, **k** is the number of unsorted planes.

PROJECT CONTINUES ON NEXT PAGE

2 Flight Dispatcher (60 Marks)

2.1 Context

A flight dispatcher assists in planning flight paths, including landing allocation. Once a plane appears on the radar, the flight dispatcher requests its number and its estimated arrival time, and stores them. The flight number and the estimated time have the same format as described in the previous question. For simplicity we limit our dispatcher system to one day.

The main mission of our flight dispatcher is to promptly allocate a landing slot to a plane - a permission for a plane to land. The landing slot must be given to a plane with the earliest estimated arrival time. If there are no planes waiting or the next plane arrives later than 5 minutes from the current time, the landing slot gets lost.

If two planes have the same estimated arrival time, the priority should be given to the plane with the plane number that comes first in alphabetical order. Once a plane is granted a landing slot, this plane should be removed from the system.

Finally, on occasion the flight dispatcher might perform an emergency landing of a certain plane.

2.2 Programming (30 Marks)

Your task is to design an appropriate data structure to store the planes and implement the constructor and all the methods of the described flight dispatcher system. Note that the method `allocateLandingSlot/allocate_landing_slot` should have $O(1)$ time complexity. In the file `Dispatcher.java/dispatcher.py`, you should implement the methods in the interface `DispatcherBase`. You may also utilise the same `Plane` class you wrote in question 1.

Your marks for this programming task will be calculated by running a series of tests against your solution using the gradescope autograder. Each test will be assigned a certain number of marks according to the rigour of the test. The total sum of all the tests will be 30 marks. The results of *some* tests will be available instantly after submission while others will be hidden until the assignment marks are released.

2.3 Report (30 Marks)

Using the gradescope document template provided, you must complete a report which analyses your code and the algorithms you have implemented. The report will be broken up into the following four sections:

1. Provide a screenshot of the functions `allocateLandingSlot/allocate_landing_slot` and `emergencyLanding/emergency_landing`.
2. Describe the data structure you used to represent a flight dispatcher.
3. State the complexities of the functions `allocateLandingSlot/allocate_landing_slot` and `emergencyLanding/emergency_landing` with respect to **n**. Briefly explain how you achieved these complexities given the data structure you have chosen.
4. If `add_plane` had to run in constant time, which alternate data structure could be used? How would this affect the running time of `allocateLandingSlot/allocate_landing_slot` and `emergencyLanding/emergency_landing`?