Module Code
COM00015C

BEng, BSc, MEng and MMath Degree Examinations 2020–2021

Department Computer Science

Title Software 1: Formative Assessment 2

Time Allowed 24 Hours (NOTE: papers late by up to 30 minutes will be subject to a 5 mark penalty; papers later than 30 minutes will receive 0 marks).

Time Recommended TWO hours

Word Limit None

Allocation of Marks:

Question 1 is worth 40% (I/O) and question 2 (Class) is worth 60%.

Instructions:

Candidates should answer **all** questions using Python 3. Failing to do so will result in a mark of 0%. All questions are independent and can be answered in any order. The solution submitted must work on the VDS, failing to do so will result in a mark of 0%.

Download the paper and the required source files from the VLE, in the "Assessment>Formative Assessments" section. Once downloaded, unzip the file. You **must** save all your code in the `ClosedExamination` folder provided. **Do not** save your code anywhere else other than this folder.

Submit your answers to the Department's Teaching Portal as a single **zip** file containing the `ClosedExamination` folder and its sub-directories. Failing to include the `ClosedExamination` folder will result in a loss of 10 marks.

If a question is unclear, answer the question as best you can, and note the assumptions you have made to allow you to proceed.

A Note on Academic Integrity

We are treating this online examination as a time-limited open assessment, and you are therefore permitted to refer to written and online materials to aid you in your answers.

However, you must ensure that the work you submit is entirely your own, and for the whole time the assessment is live you must not:

- communicate with departmental staff on the topic of the assessment
- communicate with other students on the topic of this assessment
- seek assistance with the assignment from the academic and/or disability support services, such as the Writing and Language Skills Centre, Maths Skills Centre and/or Disability Services. (The only exception to this will be for those students who have been recommended an exam support worker in a Student Support Plan. If this applies to you, you are advised to contact Disability Services as soon as possible to discuss the necessary arrangements.)
- seek advice or contribution from any third party, including proofreaders, online fora, friends, or family members.

We expect, and trust, that all our students will seek to maintain the integrity of the assessment, and of their award, through ensuring that these instructions are strictly followed. Failure to adhere to these requirements will be considered a breach of the Academic Misconduct regulations, where the offences of plagiarism, breach/cheating, collusion and commissioning are relevant: see AM1.2.1 *(Note this supercedes Section 7.3 of the Guide to Assessment)*.

1 (40 marks) Basic Programming Structure

All code for this question must be written in the file `question_1.py`, failing to do so will result in loss of marks. A test file is provided for your convenience.

Write a function `text2dictionary(filename)` where `filename` is a string containing the path to a file. The format of the file is as follow:

- each line contains the data of a single airport
- the data on each line is `airport_city, country, code`.

For example:

```
Agen, France, AGF
Paris, France, CDG
Luton, United Kingdom, LTN
```

The function must satisfy the following requirements:

- The function should return a dictionary where the keys are the country, and the values are list of tuples `(airport_city, code)`. For example:
  ```
  { 'UK' : [('Luton', 'LTN')],
    'France' : [('Paris', 'CDG'), ('Agen', 'AGF')] }
  ```
- Note, extra spaces before and after country name, airport name, and airport code must be removed.
- If the format of the file given in parameter is invalid, e.g. does not contains three entries per line, or the last entry does not contain exactly 3 characters, the function should raise a `IOError`.
- Finally, if the file contains more than once the same airport (e.g. same country, city and code), it should be added only once in the dictionary.

https://www.overleaf.com/project/5fb7d7ff3175f924323959f6

You can test your code using the file `airport_data.txt`.

2 (60 marks) Object Oriented Programming

All code for this question must be written in the file `question_2.py`, failing to do so will result in loss of marks. Four test files are provided for your convenience.

We have been asked to model different strategies of car park allocation. In order to test the initial design we will be simulating a simple strategy, park at the first available spot. The car park is a long straight line, where a car enters at one end of the straight line and exits at the other end. We have taken the design decision to store the car park spaces using a list, where empty space are represented by the value `None`, and the occupied spaces contain the vehicle `uid` (for example the matriculation plate). For simplicity we can assume that the `uid` is simply an `int`. Figure 1 shows a car park of length 10.

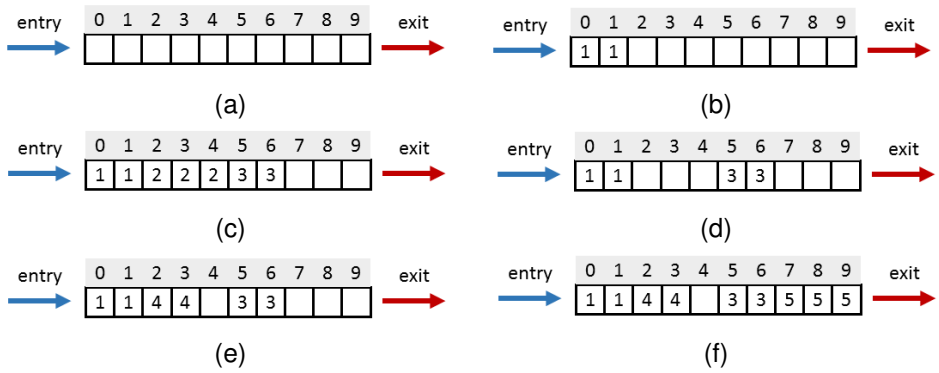

Figure 1: States of the car park after: a) initialisation, b) parking vehicle with uid 1 and length 2, c) parking vehicle with uid 2 and length 3, then vehicle with uid 3 and length 2, d) removing vehicle with uid 2, e) parking vehicle with uid 4 and length 2, and finally f) parking vehicle with uid 5 and length 3.

The results of the following sequence of events are shown in Figure 1(a-f):

(a) initialise an instance of CarPark of length 10,

(b) parking vehicle with uid 1 and length 2,

(c) parking vehicle with uid 2 and length 3, then vehicle with uid 3 and length 2,

(d) removing vehicle with uid 2,

(e) parking vehicle with uid 4 and length 2,

(f) and finally parking vehicle with uid 5 and length 3.

It should be noted that calling `get_available_space(3)` in Figure 1(d) would return 2,

not 3. In addition, in Figure 1(f) trying to park any vehicle of length greater than 1 would fail until another vehicle is removed.

(i) [15 marks] Implement the class `CarPark`'s constructor. The constructor should only take one integer parameter containing the length of the car park. The constructor should raise a `ValueError` exception if the length is less than 1. The name of the attribute containing the list of spaces **MUST BE** "`_spaces`".

(ii) [15 marks] Implement a method `get_available_spaces(position)` that takes an `int` as parameter representing a position in the car park, and returns the number of consecutive empty spaces starting from `position`. Available spaces before `position` must not be counted. The method should return 0 if no spaces are available.

(iii) [15 marks] Implement a method `park_vehicle(length, uid)` that takes two `int` as parameters. `length` is the length of the vehicle to be parked, and `uid` is the uid of the vehicle.

- The method should return the position where the vehicle is parked,
- The method should return -1 if the vehicle could not be parked, i.e. there is not enough consecutive spaces available to park the vehicle.
- The method should raise a `ValueError` exception if a vehicle with the same `uid` is already parked in the car park.

(iv) [15 marks] Implement a method `remove_vehicle(uid)` that takes an `int` for parameters representing a vehicle `uid` and removes it from the car park (i.e. replaces its `uid` by `None` in the list of spaces).

- The method should return `True` if the vehicle has been removed successfully, `False` otherwise.
- The method should return `False` if there is no vehicle with this `uid` in the car park. The method MUST NOT raise an exception in such a case.

End of examination paper