Module Code
COM00015C

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

Department Computer Science

Title Software 1

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).

Notification of errors in the paper may be made up to **one hour** after the start time. If you wish to raise a possible error it must be done through ⟨cs-exams@york.ac.uk⟩ with enough time for a response to be considered and made within the first hour.

Time Recommended THREE hours

Word Limit Not Applicable

Allocation of Marks:

Question 1 is worth 20 marks, questions 2 and 4 are worth 15 marks each, question 3 is worth 10 marks, question 5 is worth 30 marks. The remaining 10 marks are allocated to style, clarity, and quality of code.

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.

Download the paper and the required source files from the VLE, in the "Assessment>January Exam" 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. Other archival format such as .rar and .tar files will not be accepted.

If a question is unclear, answer the question as best you can, and note the assumptions you have made to allow you to proceed. Please inform ⟨cs-exams@york.ac.uk⟩ about any suspected errors on the paper immediately after you submit.

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:

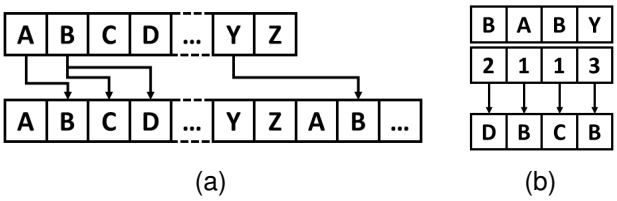
- communicate with departmental staff on the topic of the assessment
- communicate with other students on the topic of this assessment
- seek assistance with the assignment from the academic and/or disability support services, such as the Writing and Language Skills Centre, Maths Skills Centre and/or 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 (20 marks) Python Programming: Basic Programming Structure

The code must be written in the provided file `question_1.py`, failing to do so will result in loss of marks.

The Caesar cipher is a very simple encryption method and it is easily breakable. In this question we used an improved encryption method. Given a plain text message T of n characters using an alphabet A of m letters, and a list S of n positive integers from the set $\{0, 2, .., k-1\}$, T is encrypted by shifting the k^{th} character $T[k]$ by an amount $S[k]$. An example is given in Figure 1 where $T = [B, A, B, Y]$ and $S = [2, 1, 1, 3]$. The first character 'B' is shifted by 2 giving the character 'D', then the second character 'A' is shifted by 1 giving 'B' and so on. The encrypted message is "DBCB".



(a) (b)

Figure 1: Example of the encryption method proposed in this question.(a) the top row represents the character to be encrypted, the bottom row is the encrypted character given a certain shift. Note how the alphabet is repeated in the bottom row. (b) shows the encryption of the message "BABY" using a shift sequence 2113 resulting in the encrypted message "DBCB".

(i) [10 marks] Implement the function `encrypt(message, shifts, alphabet)` that takes a plain text `message` as a string, the `shifts` sequence as a list of integers, and the `alphabet` as a string. The alphabet can be any sequence of symbols, not necessarily the English alphabet. The function returns a string containing the encrypted message. The function raises a `ValueError` exception if:

- the message contains characters that are not in the alphabet,
- the size of the `shifts` is not the same as the size of `message`,
- `shifts` contains negatives values or values greater or equal to the size of the alphabet.

(ii) [10 marks] Write the **docstring** for this function. The docstring must follow Google documentation style.

2 (15 marks) Python Programming: String and List

The code must be written in the provided file `question_2.py`, failing to do so will result in loss of marks.

The aim of this exercise is to transform a string representing a molecule into a list of tuples, where the first element of a tuple is the String corresponding to an atom, and the second element is the number of occurrences of that atom at the given position in the structure of the molecule. For example, the molecule CO_2 given by the string `'CO2'` is represented by the list of tuples `[('C',1),('O',2)]`.

Glucose $C_6H_{12}O_6$ is represented by `'C6H12O6'` and `[('C',6),('H',12),('O'.6)]`. The Acetic acid molecule CH_3COOH is given by the string `'CH3COOH'` and the list of tuples `[('C',1), ('H',3), ('C',1), ('O',1), ('O',1), ('H',1)]`.

Some atoms are described by more than one character, like calcium Ca. The first character is uppercase and the second character is lower case. For example, the calcium carbonate molecule $CaCO_3$ is represented by the string `'CaCO3'`. Its list representation is `[('Ca',1), ('C',1), ('O',3)]`.

(i) [15 marks] Implement a function `molecule_to_list(molecule)` that takes a string representation of a molecule as described above and return its list of tuples representation.

The function raises a `ValueError` exception if:

- `molecule` does not start with a uppercase letter from the English alphabet,
- `molecule` contains characters that are not alphanumeric.
- `molecule` has an invalid format, for example the symbol of an atom is lower case like ca instead of Ca or h instead of H.

Up to 10 marks will be given if the function handles molecules containing only single character atoms, and up to 15 marks if the function can handle any kind of molecules.

3 (10 marks) Built-in data structures: Dictionary

The code for this question must be written in the provided file `question_3.py`. The standard atomic weight of an element is the average mass of the atoms of an element. A sample of atoms and their standard atomic weight is shown in Table 1. The molar masses of molecules are calculated from the standard atomic weights of each element. One can compute the molar mass of a compound by adding the standard atomic weights of its constituent atoms. For example the molar mass of water (H_2O) is given by:

$$\begin{aligned} m_{H_2O} &= 2 \times m_H + m_O \\ &= 2 \times 1.00797 + 15.9994 \\ &= 18.01534 \end{aligned}$$

Symbol	Name	Atomic Weight (g/mol)
H	Hydrogen	1.00797
He	Helium	4.00260
C	Carbon	12.011
N	Nitrogen	14.0067
O	Oxygen	15.9994

Table 1: List of elements and their atomic weights.

For this question, the table of atomic weight is stored in a global variable `ATOMS`. The variable `ATOMS` is a dictionary where the keys are the atoms' symbol, and the values are another dictionary. This second dictionary has two keys, the `'name'` mapped to the full name of the atom and the `'weight'` mapped to the atomic weight of that atom. An extract from the dictionary is given below.

```
ATOMS = {'H':{'name':'Hydrogen', 'weight':1.00797},
         'He':{'name':'Helium', 'weight':4.00260},
         ...}
```

A molecule is represented by a list of tuples, where the first element of a tuple is the String corresponding to an atom symbol, and the second element is the number of occurrences of that atom at the given position in the structure of the molecule. For example, the molecule H_2O is given by `[('H',2),('O',1)]`, whereas the Acetic acid molecule CH_3COOH is represented by
`[('C',1), ('H',3), ('C',1), ('O',1), ('O',1), ('H',1)]`.

(i) [10 marks] Implement a function `molar_mass(molecule)` that takes a list of tuples representing a molecule and returns its molar mass. The function must raise a `ValueError` if the molecule contains an unknown atom, that is an atom symbol that is not in the dictionary `ATOMS`.

4 (15 marks) Recursion

The code for this question must be written in the provided file `question_4.py`.

Warning: if the implemented function for this question is not recursive, a mark of 0 will be awarded. The aim of the exercise is to check if a game level is feasible or not. The game is a simple 2D platform composed of springboards with power ups and deadly traps (represented by mines). The power ups' number below the springboard represents the player's maximum jump length from that board. The aim of the game is to depart from the first springboard and arrive to the last one without stepping on a mine. Figure 2 shows a game level that is feasible, whereas Figure 3 shows a game level that a player cannot win.

Figure 2: Example of a game level that is feasible. One solution is to jump 2 steps from index 0 to 2, then 2 steps from index 2 to 4 and finally another 2 steps to the last index.

Figure 3: Example of a game level that is impossible to complete. Whatever the choice you make, you will always arrive at index 3 which is a deadly trap. It is therefore impossible to reach the last index.

To represent the level, we have chosen a list of non-negative integers. The starting point of the level is at the first index of the list, and the exit is at the last index of the list. The numbers in the list represent the power up of the springboard at that position. A mine is represented by the value 0. In this context, if the element at the last index is 0 then the level is not feasible. For example, the board shown in Figure 2 is represented by the list `[3,1,2,0,4,0,1]`. Similarly, the board shown in Figure 3 is represented by the list `[3,2,1,0,2,0,2]`.

(i) [15 marks] Implement a **recursive** function `check_level(level)` that takes a list of positive integers representing a game level and returns `True` if the level is feasible, `False` otherwise. The efficiency of the solution does not matter, you should implement a brute force approach that tries all possible cases.

5 (30 marks) User defined data structures

The code for this question must be written in the provided file `question_5.py`.

In this question, we will start building the game Blotris, a knock-off version of Tetris™. The aim is to add many random shapes to the board before we cannot add more. In Figure 4 we can see how a shape is added at a given position. When a row is fully occupied, it is removed and the rows above that row are shifted down. For example, when adding the shape in Figure 4, two rows are fully occupied. Figure 5 shows how these two rows are removed and how the rows above them are shifted down.

(i) [5 marks] Implement a class `Blotris`, with a constructor that takes two integer parameters `rows` and `cols` in that order. The parameter `rows` represents the number of rows on the board, and `cols` the number of columns. The board shown in Figure 4 was constructed with 8 rows and 5 columns.

- the class `Blotris` has an instance attribute `_board` which is a 2D list of integers representing the board. The value 0 represents an empty block, the value 1 an occupied block.
- the constructor should initialise `_board` to a 2D list of `rows` rows and `cols` columns containing only zeros.
- The constructor raises a ValueError if `rows` or `cols` is not greater of equal to 5.
- You can add more instance attributes if needed.

.

(ii) [10 marks] Implement the method `add(self, shape, row, col)` that adds a shape to a game board at row `row` and column `col`. The pair (row, col) is where the top-left corner of the shape should be. Figure 4 shows a shape added at $row = 3$ and $col = 1$. Remember rows and columns start at index 0.

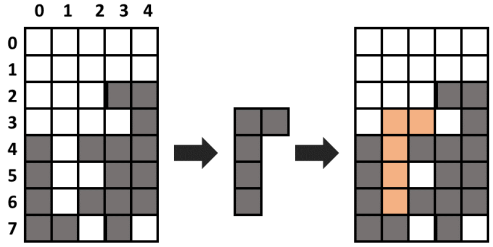


Figure 4: Example of adding a shape to the game board at row 3 column 1, that is the fourth row and the second column.

A shape is represented by a 2D list, where 0 represents an empty block and 1 an occupied block. For example, the shape added in figure 4 is represented by:

$$[[1,1],[1,0],[1,0],[1,0]]$$

- A shape can be added at a given position on the board only if it does not overlap an occupied block and is within the bounds of the board. For example, the shape in Figure 4 could not be added at $row = 3$ and $col = 0$.
- If a shape can be added, then the board is changed accordingly and the method `add` returns `True`. Otherwise, the method returns `False` and the board remains unchanged.
- If the shape is out of the boundary of the board, the method returns `False` but **must not** raise an exception.

(iii) [15 marks] Implement the method `update(self)` that updates the board, that is removes fully occupied rows and shift down the rows above accordingly. Figure 5 shows an example. In this example, rows 4 and 6 must be removed (red marks), and all rows above shifted accordingly. It can be done in several steps, first by removing the full row closest to the bottom and shifting the rows above, then removing the next full row closest to the bottom and shifting the rows above, and so on until there are no more full rows on the board. Using this algorithm, some rows may be shifted multiple times. If you prefer you can implement your own algorithm to solve the problem. It should be noted the size of the board **must not** change. The method does not return anything.

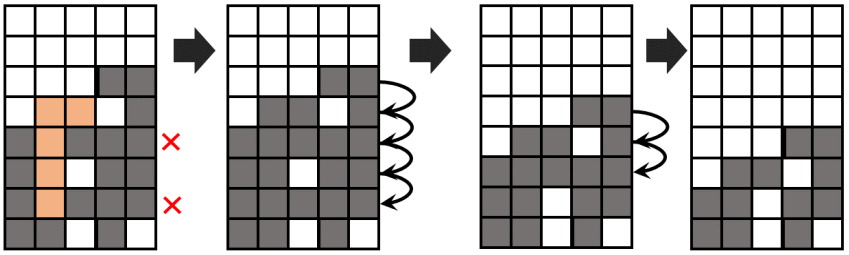


Figure 5: Example of updating the board to remove rows that are full.

End of examination paper