Module Code
COM00015C

BEng, BSc, MEng and MMath Degree Examinations 2019–2020

DEPARTMENT OF COMPUTER SCIENCE

Software 1

Sample Paper

Time Allowed:

THREE hours

Allocation of Marks:

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

Instructions:

Candidates should answer **all** questions. All questions are independent and can be answered in any order.

You should log into the computers, booted under Windows, using your special closed exam login ID (which is your exam number, e.g. Y1234123) and password (which is your username with SOF1 appended at the end, e.g. lb007SOF1). You **must** save all your code in the `ClosedExamination` directory on the `H:` drive. **Do not** save your code anywhere else other than this directory.

Materials Supplied:

Scrap paper

Do not write in any booklet before the examination begins
Do not turn over this page until instructed to do so by an invigilator

1 (15 marks) Basic Programming Structure

The code must be written in the provided file `question_1.py`.

Implement a function `to_barcode(binary)` that takes a binary string (a string composed of 0s and 1s only) as parameter and returns a string representing a bar-code. The 0s are transformed into `'.'` and 1s into `'|'`. For example:

```
>>> to_barcode('0010111')
'..|.|||'
```

In addition, the function must return `None` if the string contains a character that is not a 0 or a 1. We also assume that a string is always given, so you don't need to check the type of the parameter.

Finally, write the **docstring** for this function (5 marks). Note, the docstring must follow one of the following guidelines; PEP 0257, NumPy, or Google documentation style.

2 (15 marks) Basic Programming Structure

The code must be written in the provided file `question_2.py`.

The repetition code is one of the most basic error-correcting codes. The code repeats every data bit multiple times in order to ensure that it was sent correctly. For instance, if the data bit to be sent is a 1, an n = 3 repetition code will send 111. If the three bits received are not identical, an error occurred during transmission. If the channel is clean enough, most of the time only one bit will change in each triple. Therefore, 001, 010, and 100 each correspond to a 0 bit, while 110, 101, and 011 correspond to a 1 bit, with the greater quantity of digits that are the same ('0' or a '1') indicating what the data bit should be.

Implement a function `detect_correct(word)` with the following requirements:

- The parameter `word` is a 3-repetition code word (a string of 0s and 1s).
- The function returns a tuple containing the decoded word as first element and the number of errors in the word as second element. The decoded word is a string of 0s and 1s. For example. given the input `000111001`, the function should return `('010',1)` as the last three bits contains two 0s and one 1. This means that at least one error has occurred and the last three bits must therefore be decoded and corrected to a 0.
- The function must raise a `ValueError` if the length of the word is not a multiple of 3, or if the string contains characters other than 1 and 0.
- The function must raise a `TypeError` if the parameter word is not a string.

Finally, write the **docstring** for this function (5 marks). Note, the docstring must follow one of the following guidelines; PEP 0257, NumPy, or Google documentation style.

3 (15 marks) Built-in data structures: Dictionary

The code for this question must be written in the provided file `question_3.py`.

A two-out-of-five code is an encoding scheme which uses five bits consisting of exactly three 0s and two 1s to represent a decimal digit. This provides ten possible combinations, enough to represent the digits 0–9. This scheme can detect all single bit-errors, all odd numbered bit-errors and some even numbered bit-errors (for example the flipping of both 1-bits). The weights assigned to the bit positions are 7-4-2-1-0. For example, 2 is encoded as 00101 and 9 as 10100. However, in this scheme, zero is encoded specially, using the 7+4 combination (binary 11000) that would naturally encode 11.

Implement a function `two_out_five(message)` where:

- `message` is a string of 0s and 1s representing a number. For example the string "000111100000110" represents the number 103.
- the returned value should be a string representing the number in decimal. For example, given the string "000111100000110" as parameter, the return value is the string "103".
- if `message` is not a string, the function must raise a `TypeError`.
- if the message contains any characters that are not 1s or 0s, or has been corrupted, for example has missing digits (like "0011"), or has one or more bits flipped (that is a 1 becomes a 0 during transmission or vice-versa), the function should raise a `ValueError`.

4 (15 marks) I/O

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

The aim of this question is to implement a function that reads a table from a csv file, and store it into a dictionary before returning it. For example, the table shown in Figure 1(a) is stored in a csv file shown in Figure 1(b).

Commodity	USA	Canada	Europe	China	India	Australia
Wheat	61.7	27.2	133.9	121	94.9	22.9
Rice Milled	6.3	-	2.1	143	105.2	0.8
Oilseeds	93.1	19	28.1	59.8	36.8	5.7
Cotton	17.3	-	1.5	35	28.5	4.6

(a)

```
Commodity, USA, Canada, Europe, China, India, Australia
Wheat,61.7,27.2,133.9,121,94.9,22.9
Rice Milled,6.3, -,2.1,143,105.2,0.8
Oilseeds,93.1,19,28.1,59.8,36.8,5.7
Cotton,17.3, -,1.5,35,28.5,4.6
```

(b)

Figure 1: (a) a table in a spreadsheet, and (b) its storage in a csv file.

Implement a function `import_from_CSV(filename)` that takes a string representing the path to a file and returns a dictionary

The function must raise an IOError if the format of the file does not respect the description below.

The format of the csv file is as follow:

- The first row contains the header of each column separated by a comma. For example, the table in Figure 1 contains countries in the header of the column.
- The following rows contains the data, where the first element contains the row title (the commodity in Figure 1), followed by a comma, and then the data for each column separated by a comma.
- There is no empty line in the file.
- If there is no data in a cell, a '-' (dash sign) is used instead.
- If there is a row with too few or too many entries, the function must raise an `IOError`.

The format of the returned dictionary is as follow:

- The keys of the dictionary are the names of the countries.
- The values are dictionaries containing the data for each country. The keys of these dictionaries are names of commodities, the values are the quantity produced by that country for a given commodity. If there is no data for the given commodity (that is a dash in the csv file), the commodity **must not** be included in the dictionary. For example, cotton must not be in the dictionary for Canada. Note, a '-' (dash) is different than the value 0.

From the csv file given in Figure 1(b), a sample of the returned dictionary would be:
{'Canada':{'Wheat':27.2,'Oilseeds':19}, 'USA':{'Wheat':61.7, 'Cotton':17.3,...}, ...}

5 (30 marks) User-Defined Data Structure

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

Optimal Polygon triangulation problem.

We consider convex polygons P defined by a sequence of vertices $\langle v_0, v_1, \ldots, v_n \rangle$ where the straight line segments between consecutive vertices, called sides, form the boundary of P. Convexity means that any segment between two vertices will not go outside P. If a segment goes between two non-adjacent vertices of P it is called a *chord*. A chord splits the polygon into two smaller polygons. Note that a chord always divides a convex polygon into two convex polygons.

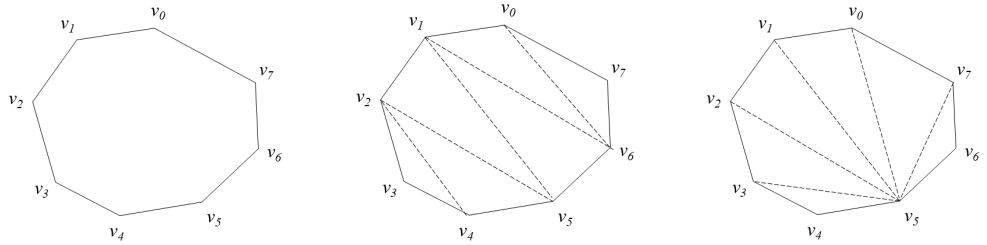

Figure 2: a) a convex polygon P, b) and c) two possible triangulations of P.

A triangulation of a polygon can be thought of as a set of chords that divide the polygon into triangles such that no two chords intersect (except possibly at a vertex). The problem is to find a triangulation that minimises the sum of the weights of the triangles. Two examples of a polygon triangulation are given in Figure 2. To determine which of the two triangulation is optimal, we need to define a cost function. The cost function used for this question is the sum of the side lengths in a triangle using the **1-norm** distance d_1 given in Equation 1.

$$d_1(v_1, v_2) = |x_1 - x_2| + |y_1 - y_2| \tag{1}$$

To facilitate your implementation, we have provided a partially implemented class `Polygon` that you will need to complete in the file `question_5.py`.

- The attribute `_vertices` stores the list of vertices representing the polygon
- The "constructor" `__init__(vertices)` creates an instance of `Polygon` given a list of vertices (a list of `Point` objects). For simplicity and in order to minimise the amount of code you need to read, we assume that the list given in the parameter represents a convex polygon.

In the same file we have also provided the class `Point` representing a point in a 2D space. The class `Point` is complete and **must not** be changed. The class has:

- two attributes `x` and `y` which are `float` representing the coordinates of the point in a 2D space.
- a static method `distance(p1, p2)` that returns the **1-norm** distance $d_1(p1, p2)$ between the two `Point` objects `p1` and `p2`.

(i) [15 marks] Implement a public method `split(p1, p2)` in the class `Polygon`. The method splits the polygon into two sub-polygons along the chord $[p1, p2]$ where `p1` and `p2` are two `Point` objects.

- The method returns a `set` of `Polygon` objects containing the two polygons resulting from the split.
- The method must raise a `TypeError` if any parameter is `None`.
- The method must raise a `ValueError` if at least one of the parameters is not a polygon's vertex.
- The method must raise a `ValueError` if `p1` and `p2` are the same vertex or two adjacent vertices in the polygon.

(ii) [15 marks] Finding the cost of an optimal triangulation of a polygon $P = \langle v_0, v_1, ..., v_n \rangle$ has a nice recursive substructure. The idea is to divide the polygon into three parts (see Figure 3(a)): a single triangle (in white), the sub-polygon to the left (dark grey), and the sub-polygon to the right (light grey). We try all possible divisions like this until we find the one that minimises the weight of the triangle plus the cost of the triangulation of the two sub-polygons.

The cost can be determine by the recursive function $cost[1, n]$. The base case of the recursion is a line segment $\langle v_i, v_{i+1} \rangle$ (that is, a polygon with zero area), which has a cost $cost[i, i] = 0$. Figure 3(a) shows the first step of the recursion, where vertex v_4 is picked for dividing the polygon P. The cost function $cost$ is defined by Equation 2.

$$cost[i, j] = \begin{cases} 0 & \text{if } i = j \\ \min_{i \leq k < j}(cost[i, k] + cost[k + 1, j] + weight(v_{i-1}, v_k, v_j)) & \text{if } i < j \end{cases} \quad (2)$$

Where $weight(v_i, v_k, v_j) = d_1(v_i, v_k) + d_1(v_k, v_j) + d_1(v_j, v_i)$ is the weight of a triangle $\langle v_i, v_k, v_j \rangle$, and $d_1(v_i, v_k)$ is the **1-norm** distance between vertices v_i and v_k.

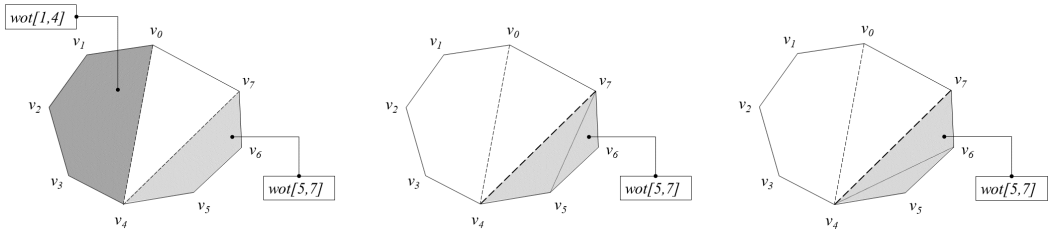

Figure 3: a) a convex polygon P, b) one possible triangulation of P, and c) an alternative triangulation of P.

For example, to triangulate the four-sided polygon $\langle v_4, v_5, v_6, v_7 \rangle$, shown in light grey in Figure 3(a), there are two alternatives:

- $k = 5 \Rightarrow cost[5,7] = cost[5,5] + cost[6,7] + weight(v_4, v_5, v_7)$ (Figure 3(b))
- $k = 6 \Rightarrow cost[5,7] = cost[5,6] + cost[7,7] + weight(v_4, v_6, v_7)$ (Figure 3(c))

If the distance from v_4 to v_6 is greater than the distance from v_5 to v_7, $k = 5$ is chosen, otherwise $k = 6$ is chosen.

Implement the public method `cost()` in the class `Polygon`. The method must return the cost (as a `float`) of an optimal triangulation of the polygon. The implementation must use recursion, failing to do so will result in a mark of 0 for this question.

End of examination paper