Computational Thinking and Programming – A.Y. 2017/2018

Written examination – 26/01/2018

Given name: _____

Family name: _____

Matriculation number: _____

University e-mail: _____

Group name: _____

Is it your first try? Yes | No

The examination is organised in three different sections:

- Section 1: basic questions [max. score: 8]. It contains four simple questions about the topics of the whole course. Each question requires a short answer. Each question answered correctly will give you 2 points.
- Section 2: understanding [max. score 4]. It contains an algorithm in Python, and you have explain what it does and to report the particular results of some of its executions according to specific input values.
- Section 3: development [max. score 4] It describes a particular computational problem to solve, and you are asked to write an algorithm in Python for addressing it.

You have 1 hour and 30 minutes for completing the examination. By the final deadline, you should deliver only the original text (i.e. this document) with the definitive answers to the various exercises that must to be written with a pen – pencils are not permitted. You can keep all the draft papers that you may use during the examination for your convenience – blank sheets will be provided to you on request.

Section 1: basic questions

1 - What is the main difference between *regular grammars* and *recursively enumerable grammars* according to the Chomsky classification of formal grammars?

2 - Consider the following two algorithms `search_1` and `search_2` – that take a list and an item as input, and return how many times the item is included in the list:

```
def search_1(my_list, item_to_search):
      result = 0
      for item in my_list:
            if item == item_to_search:
                  result = result + 1
      return result

def search_2(my_list, item_to_search):
      result = 0
      sorted_list = merge_sort(my_list)
      for item in sorted_list:
            if item == item_to_search:
                  result = result + 1
      return result
```

Explain which of the two algorithms returns the result faster than the other – and justify the answer.

3 - Select all the possible strategies for addition/deletion of elements that are used in stacks and queues:

- First In First Out (FIFO)
- First In First Abroad (FIFA)
- Last out First In (LOFI)
- Last in First Out (LIFO)

4 – Define what is a *greedy algorithm*.

Section 2: understanding

Consider the following algorithm in Python:

```
def m_cypher(list_of_chars, list_of_matriculation_numbers):
    alphabet = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
                "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"]

    list_of_numbers = []
    for number in list_of_matriculation_numbers:
        if number not in list_of_numbers:
            list_of_numbers.append(number)

    result = []
    a_len = len(alphabet)
    n_len = len(list_of_numbers)
    a_index = 0
    n_index = -1

    for char in list_of_chars:
        if char in alphabet:
            n_index = n_index + 1
            if n_index == n_len:
                n_index = 0

            a_index = a_index + list_of_numbers[n_index]
            if a_index >= a_len:
                a_index = a_index - a_len

            new_char = alphabet[a_index]
            result.append(new_char)
        else:
            result.append(char)

    return result
```

Consider the list of ten integers in your matriculation number as assigned to the variable `my_list` – for instance, if you have matriculation number *0000123456*, `my_list = [0, 0, 0, 0, 1, 2, 3, 4, 5, 6]`. What is the value returned by the aforementioned algorithm when it is called as follows:

```
m_cypher(["i", " ", "a", "m", " ", "u", "g", "o"], my_list)
```

Section 3: development

The *automated readability index* (ARI) is a readability test for English texts, designed to gauge the understandability of a text by representing the US grade level needed to comprehend such text. The formula for calculating the ARI is the following one:

$$4.71 * \frac{chars}{words} + 0.5 * \frac{words}{sentences} - 21.43$$

where *chars* is the number of letters and numbers, *words* is the number of token, and *sentences* is the number of sentences. Non-integer scores are always rounded up to the nearest whole number, so a score of 10.1 or 10.6 would be converted to 11.

Write an algorithm in Python which takes a string representing a text in input and returns the ARI for that text. As a simplification, the input text can be composed only by English characters, numbers, commas, semicolons, colons, and full stops, and no abbreviation (such as “e.g.”) can be used.