Computational Thinking and Programming – A.Y. 2018/2019

Written examination – 21/06/2019

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 – Which of the following kinds of grammars are listed in the hierarchy proposed by Chomsky:

- context-free grammars
- recursively enumerable grammars
- first recursive grammars
- low language grammars
- regular grammars

2 – Please consider the following function:

```
def f(n, s):
    if len(s) > n:
        return f(n, s[:n])
    else:
        return (n * 2) + 1
```

Write down the result returned by calling the function as follows: `f(5, "Exams!")`.

3 – Write down a small function in Python that takes in input a number and a list of numbers and return *True* if the sum of all the numbers in the input list is equal to the input number, otherwise it returns *False*.

4 – Introduce the two unordered data structures mentioned in the text book, explain their main characteristics and differences.

Section 2: understanding

Consider the following function written in Python:

```
def c(fn, mat):
    r = 0
    kind = 1
    for char in mat:
        r = r + (int(char) * kind)
        kind = kind * -1

    if r < 0:
        r = r * -1

    l = 0
    r = (r + 2) % len(fn)
    chars = list(fn)
    for idx in range(r):
        l = (l + idx) % r
        tmp = chars[idx]
        chars[idx] = chars[l]
        chars[l] = tmp

    return "".join(chars)
```

Consider the variable `my_fn` containing the **string** of your family name as written in the first page but in **lowercase**, and the variable `my_mat` the **string** of your matriculation number as written in the first page. What is the value returned by calling the function `c` as shown as follows:

```
c(my_fn, my_mat)
```

Section 3: development

Delta decoding is a way of decoding data stored in the form of differences (*deltas*) between sequential data rather than complete files. From a logical point of view the difference between two data values is the information required to obtain one value from the other. For instance, suppose to have a list of numbers, i.e. `[2, 2, 2, 3, -2]` representing the delta of a particular sequence: to calculate the decoded version of these numbers in a new list, one has to express each number in the list containing deltas as the sum of itself and its previous ones, thus obtaining the new list `[2, 4, 6, 9, 7]`. For instance, in this case, the decoded version of the third number in the list of deltas (i.e. "2") is decoded as the third number in the new list (i.e. "6").

Write a function in Python – `def delta_decoding(list_of_deltas)` – which implements the *delta decoding* for a list of integers representing deltas, and returns a new list where each number in position `x` is the decoded version, obtained following the aforementioned method, of the number in the same position in the input list of deltas.