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

Written examination – 15/05/2019

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 – What is the insertion / removal strategies that characterise either stack or queue data structures:

- First in first out (FIFO)
- First in to all (FITA)
- First in last out (FILO)
- Last in first out (LIFO)
- First in first added (FIFA)

2 – Please consider the following dynamic programming implementation of the Fibonacci function:

```
def fib_dp(n, d=dict()):
    if n not in d:
        if n == 0 or n == 1:
            d[n] = 0
        else:
            d[n-1] = fib_dp(n-1, d) + fib_dp(n-2, d)

    return d[n]
```

Correct all the errors that have been introduced.

3 – Write down a small function in Python that takes in input a string *s* and a non-negative number *n* lesser than the length of the string, and return *True* if the character of *s* in position *n* is a vowel, otherwise it returns *False*.

4 – Introduce the classes of formal grammars proposed by Chomsky and, for each class, write down the particular form that its production rules must have.

Section 2: understanding

Consider the following function written in Python:

```
def m(gr, fn, mat):
    c = ""
    gr_l = list(gr)
    fn_l = list(fn)

    if len(mat):
        idx = int(mat[0])
        gn_idx = idx % len(gr_l)
        fn_idx = idx % len(fn_l)
        n_idx = gn_idx + fn_idx

        if gr_l[gn_idx] < fn_l[fn_idx]:
            gr_l[n_idx % len(gr_l)] = fn_l[n_idx % len(fn_l)]
        else:
            fn_l[n_idx % len(fn_l)] = gr_l[n_idx % len(gr_l)]

        c = gr_l[n_idx % len(gr_l)]
        return c + m("".join(gr_l), "".join(fn_l), mat[1:])
    else:
        return c
```

Consider the variable `my_gr` containing the **string** of the name of your group as written in the first page but in **lowercase**, 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 `m` as shown as follows:

```
m(my_gr, my_fn, my_mat)
```

Section 3: development

Delta encoding is a way of storing data 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, 4, 6, 9, 7]`: calculating the deltas of these numbers means to express each number as the difference between itself and its previous one, i.e. `[2, 2, 2, 3, -2]`, where the first number of the original list is represented as it is (i.e. `2`), the second number of the original list is represented by subtracting the previous number from it (i.e. `4 - 2 = 2`), the third number of the original list is represented by subtracting the previous number from it (i.e. `6 - 4 = 2`), etc.

Write a function in Python – `def delta_encoding(list_of_numbers)` – which implements the *delta encoding* for a list of integers, and returns a new list where each number in position `x` is the encoded version, obtained following the aforementioned method, of the number in the same position in the input list.