Computational Thinking and Programming – A.Y. 2019/2020

Written examination – 10/12/2020

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 either 2 points (full answer) or 1 point (partial answer).
- Section 2: understanding [max. score 4]. It contains an algorithm in Python, and you have 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 – Select the steps that usually refers to a backtracking algorithm:

- split the input material into two or more balanced parts, each depicting a sub- problem of the original one
- if the current node is a leaf and it represents a solution to the problem, then return the sequence of all the moves that have generated the successful situation
- if the current node is a leaf but it is not a solution to the problem, then return no solution back the parent node
- reconstruct the final solution of the problem by means of the partial solutions obtained from running the algorithms on the smaller parts of the input material
- return the solution calculated previously to the problem if this is the case

2 – Identify and correct the mistake(s) in the following greedy algorithm:

```
def line_wrap(text, line_width):
    result = list()
    space_left = line_width
    line = list()
    for word in text.split("."):
        word_len = len(word)
        if word_len + 1 > space_left:
            result.append(" ".join(line))
            line = list()
            line.append(word)
            space_left = line_width - word_len
        else:
            line.append(word)
            space_left = space_left - word_len + 1
    return "\n".join(result)
```

3 – Write down a small function in Python that takes in input a boolean and an integer and returns `True` if the input boolean is `True` and the input integer is an even number, while it returns `False` otherwise.

4 – What is the main difference between *regular* grammars and *context-free* grammars according to the Chomsky classification of formal grammars?

Section 2: understanding

Consider the following functions written in Python:

```
def f(last_name, mat):
   result = None

   d = {}
   i = -1
   for c in last_name:
       if c not in d:
           i = i + 1
           d[i] = c

   if len(d) > 0:
       for n in mat:
           i = int(n) % len(d)
           return d[i] + f(last_name[1:], mat[1:])

   return "$"
```

Consider the variables `my_family_name` containing the string of your family name and the variable `my_mat_number` containing the **string** of your matriculation number (ten digits). What is the value returned by calling the function `f` as shown as follows:

```
f(my_family_name, my_mat_number)
```

Section 3: development

The **Coleman–Liau index** (**CLI**) 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 CLI is the following one:

$$CLI = 0.0588 * L - 0.296 * S - 15.8$$

where, considering a given text, *L* is the average number of alphabetic characters and digits (no punctuation and spaces are considered here) per 100 words and *S* is the average number of sentences per 100 words.

Write an algorithm in Python – `def cli(text)` – which returns the CLI of the text specified in input.