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

Written examination – 26/02/2018

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 - Describe what are the components of a Turing Machine.

2 - Consider the following function, that takes as input only (negative or positive) numbers:

```
def fun(n):
      if n < 0:
            return n
      else:
            return fun(n+1)
```

Does this function terminate for any possible input number? Justify your answer.

3 - Select the steps that usually refers to a backtracking algorithm:

- [leaf-win] 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
- [divide] split the input material into two or more balanced parts, each depicting a sub-problem of the original one
- [leaf-lose] if the current node is a leaf but it is not a solution to the problem, then return no solution back the parent node
- [recursive-step] apply recursively the whole approach for each child of the current node, until one of these recursive executions returns a solution – if none of them provides a solution, return no solution back the parent node of the current one
- [combine] 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

4 – Define what is an *algorithm*.

Section 2: understanding

Consider the following functions written in Python:

```
def w_count(s, text):
    result = {}

    c_values = {}
    for c in s.lower().replace(" ", ""):
        if c not in c_values:
            result[c] = 0
            c_values[c] = 0
        c_values[c] = (c_values[c] + 1) * 2

    for k in c_values:
        result[k] = calculate(k, c_values[k], text.split())

    return result


def calculate(key, value, token_list):
    l_len = len(token_list)
    if l_len == 0:
        return 0
    else:
        cur_token = token_list[0]

        if key in cur_token:
            result = value
        else:
            result = -1

        return result + calculate(key, value, token_list[1:l_len])
```

Consider the variable `name` containing the string of your full name (i.e. given name plus family name separated by a space) as you have specified in the first page of this examination document. What is the value returned by calling the function `w_count` as shown as follows:

```
w_count(name, "Begin at the beginning and go on till you come to the end: then stop.")
```

Section 3: development

The *Hamming distance* between two strings of **equal** length is the number of positions at which the corresponding characters are different. Thus, it measures the minimum number of substitutions required to change one string into the other.

Write an algorithm in Python which takes two strings as input and that calculates the Hamming distance if the strings have the same length, otherwise it returns the smallest string.