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

Written examination – 15/03/2019

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 boolean expressions return `True` as final value:

- `True or False`
- `not False and (True or False)`
- `True and not True`
- `not (False or True or False)`
- `True and not (True and False)`
- `not (False and not True) or False`

2 – Please consider the following function:

```
from collections import deque

def f(s1, s2):
    l = list()
    indexes = deque(range(len(s1)))
    while len(indexes) > 0:
        idx = indexes.popleft()
        if idx < len(s2):
            l.append(s2[idx])
        else:
            l.append(s1[idx])
    return "".join(l)
```

What is the value returned by executing it as follows: `f("big", "brother")`

3 – Write down a small function in Python that takes in input two strings and returns *-1* if the first string is longer than the second string, *0* if the strings have the same length, and *1* if the second string is longer than the first string.

4 – Introduce the problem of the seven bridges in Königsberg and sketch out (in natural language) the resolution of such problem that was provided by Euler.

Section 2: understanding

Consider the following function written in Python:

```
from collections import deque


def f(fn, mn):
    l = deque()
    digits = []

    for i in range(len(fn)):
        j = i % len(mn)
        digits.append(int(mn[len(mn) - 1 - j]))

    for idx, d in enumerate(reversed(digits)):
        if idx < (len(digits) / 2):
            l.append((d, digits[idx]))

    result = []
    for c in fn:
        result.append(c)

    while l:
        t = l.pop()
        if t[0] < len(fn) and t[1] < len(fn):
            tmp = fn[t[0]]
            result[t[0]] = fn[t[1]]
            result[t[1]] = tmp

    return "".join(result)
```

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_mn` the **string** of your matriculation number as written in the first page. What is the value returned by calling the function `f` as shown as follows:

```
f(my_fn, my_mn)
```

Section 3: development

The **Rabin–Karp algorithm** is a very efficient approach, used for detecting plagiarism, that allows one to find whether a particular *pattern* string is contained in another *input* string. Instead of comparing character by character the pattern with the string, it computes the *hash* value of the pattern and compares it with the hash value of each substring of length equal to pattern. The hash value is calculated by means of an hash function that assigns a number to a particular string, so as to make number comparisons instead of character comparisons, which are less efficient. Of course, multiple executions of the hash function on the same string returns always the same number.

In practice, starting from the beginning of the input string, the algorithm compare the hash value of the pattern with the hash value of the *k* characters of the input string starting from the first one, where *k* is the length of the pattern string. If the two values do not match, then it compares the hash value of the pattern with the hash value of the *k* characters of the input string from the second one. And so on, until either it found a match or no matches were found.

Write a function in Python – `def rabin_karp(input, pattern)` – which implements the *Rabin–Karp algorithm* technique and returns True when a match is found, or False otherwise. The hash value can be calculated by means of the Python built-in function `def hash(s)`, which takes a string in input and returns an integer representing the hash value of the input string.