15-110 Fall 2019
Homework Exam 01

Out: Sunday 17th November, 2019 at 15:00 AST
Due: Sunday 17th November, 2019 at 16:20 AST

Introduction

This homework exam includes questions selected from previous homework assignments.
The total number of points available from the questions is 120, where 20 points are *bonus* points (i.e., you only need 100 points to get the maximum grade).

Warning: if the code of a function doesn't execute because of syntax errors of *any type* (i.e., the function doesn't even reach the end) then you'll get 0 points, the function won't be evaluated at all during the grading process! This means that you should / must try out your code, function by function, before submitting it!

In the handout, the file `hwexam01.py` is provided. It contains the functions already defined but with an empty body (or a partially filled body). You have to complete the body of each function with the code required to answer to the questions.
You need to submit to Autolab the `hwexam01.py` file with your code.

Only the provided *reference cards* (possibly with your annotations) are admitted as a support during the exam.

The code must be written and tested using Spyder on the computers in the classroom.

1 Count the occurrences of a substring

Problem 1.1: (27 points)

Implement the function `count_occurrences(s, ss)` that takes as inputs two strings, `s` and `ss`. The purpose of the function is to check whether the substring `ss` is contained in the string `s`.

As an example, let's assume that `s` is the string `'Mountains, sea, lakes. Sea with green waters.'` and `ss` is the string `'sea'`.

- If `ss` is a substring of `s`, the function returns a multi-line output string. In the case of the example case above, the output would be as follows:

  ```
  ''The substring 'sea' has been found 2 times.
  The first occurrence is at position 11.
  The string content following the last occurrence of 'sea' is ' with
      green waters.''
  ```

  Note that the output must be a *multi-line string* (more precisely, a string that would print out over three lines).

  Note also that the substring can be found in *any lower/upper case combination*. E.g., both `'Sea'` and `'sea'` count.

- If `ss` is not a substring of `s`, the function returns the following substring: `''The string 'sea'is not part of the input string''`, where of course `'sea'` should be replaced by the value of the given string `ss`.

2 Shifting list elements

Problem 2.1: (23 points)

Implement the function `slice_shift(l, pos_from, pos_to)` that takes as inputs a list `l` and two integers. The function *shifts* the two elements at positions `pos_from` and `pos_from + 1` to the position `pos_to` in the list. The new list, with the shifted items, is returned. This list must have a different identity compared to `l`.

For instance, `slice_shift([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], 4, 2)` returns
`[1, 2, 5, 6, 3, 4, 6, 8, 9, 10, 11]`

3 Decimal to binary

Problem 3.1: (21 points)

Implement the function `decimal_to_binary(d)` that takes an integer `d` as input, and returns the string of zeros and ones that is the binary representation of `d`.

4 Friends

Consider a dictionary that stores a list of people that consider each other friends:

```
d = dict()
d["fred"] = set(["betty", "barney"])
d["wilma"] = set(["fred", "betty"])
d["betty"] = set(["barney"])
d["barney"] = set()
```

In this example, fred considers his friends to be betty and barney; wilma considers her friends to be fred and betty; betty considers her only friend to be wilma, and barney believes he has no friends.

Problem 4.1: (22 points)

Write the helper function `likes(d, person)` that takes a dictionary of the given form, and a person you may assume is in the dictionary, and returns the set of people who list that person as a friend.

5 Grades

Now that you know how files work, suppose we are keeping track of students' grades in a file that looks like this:

```
rick  , 10, 10 , 10 , 9.7, 8.7
morty , 8 , 7.5, 10 , 9  , 7.6
beth  , 7 , 9.6, 8.5, 10
jerry , 6 , 5.4, 3.8, 10
summer, 10, 9.5, 8.5, 5  , 7.2, 8
```

Each student is on a separate line, where the first element is their name, followed by their scores, separated by spaces. Notice that not all students have the same number of assignments.

Problem 5.1: (27 points)

Implement the function `compute_grade(filename)` that takes as input the name of the file that contains the grades, and writes a file named `grades.txt` where each line corresponds to a student, formatted as follows:

- the name of the student as a string of 15 characters;
- the average grade of the student as a floating point number of 5 digits, where 2 correspond to the decimal places;
- the letter grade A, B, C, D, or R, depending on the average x of the student:
  - A if $x \geq 9$
  - B if $9 < x \geq 8$
  - C if $8 < x \geq 7$
  - D if $7 < x \geq 6$
  - R otherwise

The function should return the string that was written in the file.