CSC 121
Computational Assessment Tasks 7

Spring 2020

Please read these instructions carefully:

- You must complete this CAT by noon (EST) on Tuesday, March 30, 2020, **or** within 1.5 hours of first reading this document, **whichever is sooner**. If you are reading this, then the clock is already ticking.
- There are **two** programming problems worth a total of **17** points.
- You may use the following resources: a calculator, your class notes, the course textbook, any code you have written for this class, content on the CSC 121 Moodle page (lecture slides, code samples, homework solutions), and the Python tutorial and library references pages: `https://docs.python.org/3.6/index.html`
- You may *not* use any resources other than those specifically listed above — you may not work with others, consult online forums, use search engines etc. This is a pledged activity and the Davidson Honor Code applies.
- There is a single electronic deliverable — `CAT_7.py` — that is to be turned in via Gradescope. Don't forget to also sign your name on Moodle when you complete the CAT.
- Your submission must follow the code style guidelines for the class. If you are unable to complete a problem, submit your partial solution including comments describing your overall approach to the problem and where you are stuck. Such documentation will be factored into partial credit computations.

Getting Started

Begin by downloading the supplementary archive file from Moodle. This archive file contains a skeleton file named `CAT_7.py` as well as a sample input file for the Cheeseburger problem. Your task is to implement the indicated functions in the skeleton file.

[5pts] Cheeseburger

Tere loves cheeseburgers — so much so that her first act when she goes to a new restaurant is to scan the menu to determine how many different types of cheeseburgers are listed. On this problem, your task is to write a function to help out Tere. Specifically, complete the body of the function named `count_cheeseburgers` that:

- accepts the name of a file as an input parameter,
- computes the number of lines in the file that begin with the word `cheeseburger` and
- returns this count as an integer.

Your function should not print any output to the screen.

Notes and Constraints

- You may assume that the specified file will exist in the same folder as your module and will be readable. Thus, you don't have to use a `try-except` statement to trap any `IOError`s.
- You should only count those lines that begin *precisely* with the string `cheeseburger` — for example, we will consider the plural or capitalized form of this string, or one with embedded punctuation, to be different. Thus, you should not count those lines where the first the word is `cheeseburgers`, `Cheeseburger`, `cheeseburger!`, etc.
- The input file may contain blank lines.

Example

A sample input text file, `file1.txt`, is provided to you as part of the supplementary archive file. The function call `count_cheeseburgers("file1.txt")` should return 3. Only lines 1, 6, and 7 in `file1.txt` begin precisely with the word `cheeseburger`, so those are the only ones that are counted.

[12pts] Minimum Cost Palindrome[1]

A palindrome is a string that reads the same from left to right as it does from right to left. You are given a string `string_to_change` of even length. Each character in this string is either an `'o'`, an `'x'`, or a `'?'`. The goal of this task is to replace each occurrence of `'?'` with either an `'o'` or an `'x'` so that `string_to_change` becomes a palindrome. Further, you need to do this in the cheapest way possible. You are given integers `o_cost` and `x_cost` that denote the cost of replacing a `'?'` with an `'o'` and an `'x'` respectively. Complete the body of the function named `min_cost_palindrome` that given `string_to_change`, `o_cost` and `x_cost` returns the minimum cost of replacing `'?'`s with `'x'`s and `'o'`s that turns `string_to_change` into a palindrome. If it is impossible to obtain a palindrome, your function should return -1 instead. Your function should not print any output to the console.

Notes and Constraints

- You are not allowed to change an `'x'` into an `'o'` or vice versa.
- You are guaranteed that the length of `string_to_change` will be even. Note that the empty string has even length (length 0) and is a palindrome.
- Every character in `string_to_change` will be either an `'x'`, an `'o'` or a `'?'`.
- If you write a helper function, remember to write a docstring for it!
- **Hint:** Note that we are only interested in computing the *cost* of transforming `s` into a palindrome — you can perform this computation without actually trying to turn `s` into a palindrome.

[1]This problem statement is the exclusive and proprietary property of TopCoder, Inc.

Examples

1. `string_to_change = "oxo?xox?", o_cost = 3, x_cost = 5`
   Returns 8. The only way to produce a palindrome is to replace `string_to_change[3]` with an `'x'` and `string_to_change[7]` with an `'o'`. The first replacement costs 5 and the second 3, for a total of 8.

2. `string_to_change = "x??x", o_cost = 9, x_cost = 4`
   Returns 8. There are two ways to produce a palindrome here. The cheaper option is to change both `'?'`s into `'x'`s. This costs $4 + 4 = 8$. Note that you are required to replace all `'?'`s.

3. `string_to_change = "ooooxxxx", o_cost = 12, x_cost = 34`
   Returns -1. There is no `'?'` character in `string_to_change` and it's not already a palindrome.

4. `string_to_change = "oxoxooxxxxooxoxo", o_cost = 7, x_cost = 4`
   Returns 0. There is no `'?'` character; however, `string_to_change` is already a palindrome and making no replacements costs nothing.