Practice Arena

Practice problems aimed to improve your coding skills.

- PRACTICE-02_SCAN-PRINT
- PRACTICE-03_TYPES
- LAB-PRAC-02_SCAN-PRINT
- LAB-PRAC-01
- PRACTICE-04_COND
- BONUS-PRAC-02
- LAB-PRAC-03_TYPES
- PRACTICE-05_COND-LOOPS
- LAB-PRAC-04_COND
- LAB-PRAC-05_CONDLOOPS
- PRACTICE-07_LOOPS-ARR
- LAB-PRAC-06_LOOPS
- LAB-PRAC-07_LOOPS-ARR
- LABEXAM-PRAC-01_MIDSEM
- PRACTICE-09_PTR-MAT
- LAB-PRAC-08_ARR-STR
- PRACTICE-10_MAT-FUN
  - Super Getline
  - Tailor-made Tabs
  - De-Duplicate
  - Matrix Mania
- LAB-PRAC-09_PTR-MAT
- LAB-PRAC-10_MAT-FUN
- PRACTICE-11_FUN-PTR
- LAB-PRAC-11_FUN-PTR
- LAB-PRAC-12_FUN-STRUC
- LABEXAM-PRAC-02_ENDSEM
- LAB-PRAC-13_STRUC-NUM
- LAB-PRAC-14_SORT-MISC

Tailor-made Tabs

PRACTICE-10_MAT-FUN

The anatomy of the tab character is a bit funny. It does not always produce a sequence of 4 spaces. Instead, it produces just enough spaces so that the next character after the tab character is aligned to what is known as a *tab stop*. See the following websites for details
https://en.wikipedia.org/wiki/Tab_key (https://en.wikipedia.org/wiki/Tab_key)
https://en.wikipedia.org/wiki/Tab_stop (https://en.wikipedia.org/wiki/Tab_stop)

In this question we will implement a new tab character where the user can specify the tab size. The first line of the input will contain a strictly positive integer n and the second line will contain a string. However, the string may contain escape sequences. You have to reprint the string realizing the escape sequences yourself. The escape sequences you have to handle are:

\n Newline
\\ Backslash
\" Double quotation mark
\T Tailor-tab (described below)

If the character \ (backslash) is followed by any other character, print the word "ERROR" (without quotes) as your output.

For example, if the input string is #ABC\"DEF\"# (without the pound symbols), then your output should be
ABC"DEF"

There will be no traditional tab (\t) character in the string. However, in addition to the above escape sequences, you have to handle a new escape sequence tailor-tab that we define below. Tailor-tab \T is a new tab character we are defining here that aligns characters to the next column divisible by n (n was given to you in line 1). For example if n = 3 and the string is #ABC\TDEF# (without the pound symbols), then your output should be

```
ABC  DEF
```

Note that there are 2 spaces after the letter C so that the next character D gets printed in column 6 which is divisible by 3 (D would have otherwise gotten printed in column 4 which is not divisible by 3). Note that newline characters refresh the column count. Thus, if n = 3 and the user string is #ABC\n\TDEF# (without the pound symbols) then the output must be

```
ABC
  DEF
```

so that the character D gets printed in column 3 which is divisible by 3 (it would have otherwise been printed in column 1 which is not divisible by 3.

The user input string will be given in a single line and contain no more than 499 characters. We also assure you that the total characters you have to output (including the extra spaces) will not exceed 999.

Bonus: implement a new function tailorTabs that can take a string and the tab size as its two arguments and print that string according to the rules mentioned above.

You should be able to solve this problem nicely even without using functions. Functions will be covered in class next week. Functions allow you to write cleaner code and allow you to think about the problem in an organized manner. However, although it is perfectly possible to solve most problems in a reasonable manner without using functions at all, code (especially if it is long and complex) written without functions, tends to be more error prone and hard to debug.

Start Solving! (/editor/practice/6179)