








































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_CONDDLOOPS
-  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
 -  Il fratello di Fibonacci
 -  Hidden Palindrome
 -  Hush Hush Hash
 -  Maximum Match
 -  El secreto de sus l
 -  Star Replacement
 -  Stronger together
 -  Rigorous and repeated redaction
 -  strnrev
 -  Monster and Mini Multiply
 -  Clash of the Substrings
 -  Personalizing Emails
-  PRACTICE-10_MAT-FUN
-  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

Hush Hush Hash

LAB-PRAC-08_ARR-STR

Hush Hush Hash [20 marks]**Problem Statement**

The concept of a hashing is critical in computer science. Hashing is the art and science of converting a given piece of data, like a string, into a single number which can be stored. For instance, the CC (or even GMail, Piazza, Gradescope etc) does not store your password anywhere. It just hashes your password stores the hash. When you try to logon to CC Email and enter your password, the CC hashes what you have entered and compares this hash value and compares it to the hash value that it has stored. If the two match, you are given login otherwise, you are denied login.

The CC etc use sophisticated and cryptographically secure hash functions like SHA etc. In this question, we will design two very simple hash functions. Your input will be a string with n characters where n will be equal to 499 or less. We promise that the characters will either be English letters 'A', 'x', 'E' etc or else digits '0', '6' etc. You have to take this string and convert this into a number as follows:

Assign every character a number in the following manner

1. Digits get assigned their own value i.e. '0' gets assigned value 0, '6' gets assigned value 6 etc.
2. Letters get assigned a value equal to their position in their alphabet. E.g. 'A' and 'a' both have value 1, 'C' and 'c' both have value 3, 'Z' and 'z' both have value 26 etc.
3. Note that the values as calculated above are always non-negative integers.

Suppose your string is sitting inside a character array called `str` with the n characters stored in `str[0]` to `str[n-1]` and `str[n]` being the NULL character '\0' which Mr C should himself put in if you read the string from the input properly. Calculate the following two hash values for this string. For a character x let $val(x)$ denote its value as calculated above.

$$hash1(str) = \sum_{i=0}^{n-1} val(str[i])$$

$$hash2(str) = \left(\prod_{i=0}^{n-1} val(str[i]) \right) \% 100000$$

Print the two hash values calculated above (they will always be non-negative integers), in two separate lines in your output.

Caution

1. Be careful about extra/missing lines and extra/missing spaces in your output.
2. Once you have gone back home after finishing today's lab, you may want to look up the term "Hash function" on the internet (don't browse the internet during the lab though - it is banned)

HINTS:

1. The product of 499 numbers may be very large (and may overflow the limit of int/long) but the hash value in hash2 is always between 0 and 99999. Use the modulus trick we earlier saw that $(a * b) \% c = ((a \% c) * (b \% c)) \% c$
2. You may want to use the strlen function from string.h to find out the length of a string

EXAMPLE 1:

INPUT

1234

OUTPUT:

10

24

EXAMPLE 2:

INPUT

AbCd

OUTPUT:

10

24

Grading Scheme:Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries 50% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

 **Start Solving!** (/editor/practice/6165)