








































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_CONDLLOOPS
-  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
-  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
 -  Too tired to create a story - part I
 -  Too tired to create a story - part II
 -  Too tired to create a story - part III
 -  Point Proximity
 -  The Bisection Method
 -  The pace is too fast
 -  A Question on Quadrilaterals
 -  The Trapezoidal Technique
 -  Constrained Candy Crush
 -  Major Mobile Madness
 -  The Newton Raphson Method
 -  The Palindrome Decomposition
-  LAB-PRAC-14_SORT-MISC

The Palindrome Decomposition

LAB-PRAC-13_STRUC-NUM

The Palindrome Decomposition [20 marks]

Problem Statement

Any string can be decomposed into a sequence of palindrome strings (since a single letter is always a palindrome). For example, the string `abc` can be decomposed as `[a][b][c]` which we will represent as `123` denoting the fact that `[a]` is the first palindrome, `[b]` is the second palindrome and `[c]` is the third palindrome. On the other hand, `abab` can be decomposed in the following three ways (the corresponding numerical representations are also given).

`[aba][b] => 1112`

`[a][bab] => 1222`

`[a][b][a][b] => 1234`

You will be given a string with 9 or less characters (all characters will be English lowercase alphabets). You have to print the numerical representations of all possible palindromic decompositions of the string in the output. Each decomposition should be printed on a separate line with no spaces anywhere. Notice that each decomposition seems like a number with k digits where k is the length of the string we gave you. Print these numerical representations in increasing order

Caution

1. Even if the same palindrome repeats in the string, use a different counter to denote its occurrence in the decomposition. For instance, you may decompose the string `abaaba` as `[aba][aba]`. However, this will correspond to the representation `111222` and not `111111` since the second `aba` is a part of a different palindrome and not the first palindrome.
2. The first palindrome in the decomposition is to be denoted the number 1, not the number 0.
3. We will not penalize you for extra new lines at the end of your output but do not have extra spaces anywhere in your output.

HINTS: Write a recursive function `printPalinDecomp(char *str, int *decomp, int len, int done, int counter)` to solve this problem.

1. `str`: string given as input
 2. `decomp`: an integer array to store the numerical representation of the decomposition
 3. `len`: length of the string
 4. `done`: how many positions of the string have we processed so far?
 5. `counter`: how many palindromes have we generated so far?
-

EXAMPLE:

INPUT

`abab`

OUTPUT:

`1112`

`1222`

1234

Explanation: see above.

Grading Scheme:

Total marks: **[20 Points]**

There will be partial grading in this question. There are several lines in your output. Printing each line correctly, in the correct order, carries equal 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/6266)**