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
- LAB-PRAC-09_PTR-MAT
- LAB-PRAC-10_MAT-FUN
- PRACTICE-11_FUN-PTR
- LAB-PRAC-11_FUN-PTR
  - Name the Clones
  - The Race of the Clones
  - Partial Palindrome
  - Growth Curve
  - The Family Tree of Mr C
  - Timely Tasks
  - Plenty of Palindromes
  - Count and Say Sequence
  - Orbiting Indices
  - Zig-zag Numbers
  - Parent Palindrome
  - Leaderboard
- LAB-PRAC-12_FUN-STRUC
- LABEXAM-PRAC-02_ENDSEM
- LAB-PRAC-13_STRUC-NUM
- LAB-PRAC-14_SORT-MISC

Plenty of Palindromes

LAB-PRAC-11_FUN-PTR

Plenty of Palindromes [20 marks]

Problem Statement
In the input, you will be given two strictly positive integers n and k, separated by a space. You have to generate all possible strings that satisfy the following properties

1. The strings should all be of length k
2. The strings should only contain the first n lower-case English alphabet characters. However, the allowed characters may repeat any number of times (some characters may be absent as well).
3. The strings should all be palindromes

Print each palindrome string on a different line of your output. You have to print the strings in *lexicographically* increasing order, i.e. the order in which these strings would appear in a dictionary. We explain lexicographic ordering below. Thus, you have to first output the lexicographically smallest palindrome string of length k you can construct out of the first n lower-case letters of the English alphabet. Then find the lexicographically next smallest palindrome string of length k and so on till you have generated all such palindromes possible.

Lexicographic ordering
Just as given two digit sequences, say 1923 and 3122, we can say which digit sequence is "smaller" and which is "larger" (by interpreting the digit sequences as numbers), two sequences of alphabets, i.e. strings, can also be compared and given two sequences of alphabets, we can say which one is smaller and which one is larger.

The rules of doing so are pretty simple if the two strings are of the same length, which will be the case in this question. We first declare that the character 'a' is smaller than the character 'b', which is in turn smaller than the character 'c' and so on. To compare two strings, simply look at their first characters - the string with the larger character wins. If both strings have the same first character, then the second characters of the two strings are compared and so on.

Thus, we have "cat" = "cat" since the two strings are exactly the same but we have "cap" < "cat" since 'p' is smaller than 't'. Also, we have "mat" > "cat" since 'm' is larger than 'c' and also "aazd" < "abbb" since 'a' is smaller than 'b' (The third characters 'z' and 'b' do not get compared at all since "abbb" wins when the second characters get compared).

Caution

1. Using recursion is not compulsory in this question. However, you will have to write much less code if you use recursion.
2. Do not output the strings in incorrect order. The autograder will heavily penalize you if you do this since it will give you marks only if a correct string is output in its correct location.
3. We will not penalize you for stray newlines at the end of your output. However, do not have stray spaces at the end of each line of your output. You will not pass test cases if you have these.

HINTS:

This problem may seem very complicated if you try to write a loop to solve the problem but it can be very elegantly solved using recursion as we explain below. Suppose we want to generate all palindromes of length 4 out of the first two alphabet letters, in lexicographic order. This can be solved if we first generate all palindromes starting with a and then generate all palindromes starting with b (since strings starting with a come before those starting with b in lexicographic order).

However, if a palindrome starts with an a, it must also end with an a. This means that all such palindromes must be of the form a**a where ** is a palindrome of length 2. Thus, we can solve this problem using a smaller version of the original problem which is why recursion can be used (to generate all palindromes of length 4, we need all palindromes of length 2).

Write a function that takes partially filled in palindromes and recursively calls itself to complete the palindromes. Write a function generatePalindromes(char* str, int k, int n, int next) which takes in four arguments

1. a character array name of length k+1 (k characters for the names and one for the NULL character)
2. the value of k (will help you know what is the length of the array)
3. the value of n (which all letters of the English alphabet can you use)
4. the first unfilled position in the array (denoted by * in the above example)

The base case of the recursion can be when only one or two positions are left unfilled in the string (depending on whether k is odd or even). In this base case, we simply need to loop through all valid characters and use them to fill in the empty positions. To start off, you can invoke this recursion using something like the following

```
char str[k+1];
str[k] = '\0';
generatePalindromes(str, k, n, 0);
```

where next = 0 indicates that the array is completely empty at this point.

Use these hints to completely solve the problem

EXAMPLE 1:
INPUT
2 4

OUTPUT:
aaaa
abba
baab
bbbb

EXAMPLE 2:
INPUT
2 5

OUTPUT:
aaaaa
aabaa

ababa
abbba
baaab
babab
bbabb
bbbbb

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/6221)