








































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
-  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

Zig-zag Numbers

LAB-PRAC-11_FUN-PTR

Zig-zag Numbers [20 marks]

Problem Statement

A number is called a zig-zag number if the second digit of the number is strictly greater than the first, the third digit of the number is strictly smaller than the second, the fourth digit is strictly smaller than the third, and so on. Some examples include 121212, 1425 etc.

In the input, you will be given two strictly positive integers n and k , separated by a space. You will have to generate all possible zig-zag numbers that have exactly k digits in them, using only the digits $1, 2, \dots, n$. You must print all the possible zig-zag numbers in increasing order. A digit may repeat any number of times in a zig-zag number but the zig-zag rule must be followed.

We assure you that n will be strictly greater than 1 but strictly less than 10 (i.e. it will be a single digit number but not 1 or 0). To make this question a bit easier for you, we also assure you that k will be always an even number.

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 numbers in incorrect order. The autograder will heavily penalize you if you do this since it will give you marks only if a correct number is output at 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. It is easy to see that there can be no zig-zag numbers using a single digit. Let us try to construct all zig-zag numbers with two digits. It is again easy to see, that if $n = 2$, then there is only one zig-zag number with 2 digits i.e. 12. If $n = 3$, then we have more two-digit zig-zag numbers, namely 12, 13 and 23. If $n = 4$, then we have the possible two-digit zig-zag numbers as 12, 13, 14, 23, 24.

This forms the simplest case of a recursive definition of zig-zag numbers. Suppose we want to generate all zig-zag numbers with 8 digits starting with 15, then notice that this problem can be solved simply by appending to 15, all possible zig-zag numbers with 6 digits. However, there is a tiny restriction here that these 6 digit zig-zag numbers must start with a digit that is strictly smaller than 5.

Write a function that takes partially filled in zig-zag numbers and recursively calls itself to complete the numbers. Write a function `generateZigZag(char* num, int k, int n, int left, int max)` which takes in five arguments

1. a character array name of length $k+1$ (k characters for the number 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 digits can be used in the number)

4. left: how many digits are left to be filled in?
5. max: what is the maximum value that the first digit in the remaining number can take

The base case of the recursion can be left = 0 which means we have a complete number which can simply be printed. To print all zig-zag numbers using digits 1,2, ..., n starting with the "15", you could do something like the following:

```
char num[k+1];  
num[k] = '\0'; // Do not forget the NULL character  
num[0] = '1';  
num[1] = '5';  
generateZigZag(num, k, n, k - 2, 4) // two digits already filled, k-2 left to be filled, next digit must be  
strictly smaller than 5
```

Use these hints to completely solve the problem.

EXAMPLE 1:

INPUT

2 4

OUTPUT:

1212

Explanation: 1212 is the only four-digit zig-zag number possible using the digits 1,2.

EXAMPLE 2:

INPUT

3 4

OUTPUT:

1212

1213

1312

1313

1323

2312

2313

2323

Explanation: Notice that all the numbers are placed in increasing order.

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/6224)**