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
- LAB-PRAC-12_FUN-STRUC
- LABEXAM-PRAC-02_ENDSEM
  - Meanie Numbers
  - Rotate Then Rotate Code
  - The enigma that was Enigma
  - Save the Date
  - Pretty Patterns
  - Trivial Tic-Tac-Toe
  - How Mr C reads your code
  - Malloc Mystery
- LAB-PRAC-13_STRUC-NUM
- LAB-PRAC-14_SORT-MISC

The enigma that was Enigma

LABEXAM-PRAC-02_ENDSEM

The enigma that was Enigma [marks]

Problem Statement
At the height of World War II, in the early 1940s, British mathematicians, led by Dr. Alan Turing, developed a code breaker machine called Enigma at Bletchley Park. Enigma could easily catch and breakdown the messages being sent from and to the Axis powers. Mr C is inspired by this achievement and wants to do the same.

The first line of the input will give you a strictly positive number n. The next line will give you n digits i.e 0, 1, 2, ..., 9 (no spaces). These digits actually stand for a secret message but there could be many such secret messages, as described below. Your job is to print all secret messages in lexicographic order, i.e. in the order these messages would appear in a dictionary.

Suppose n = 4 and the 4 digits are 1234. Then we can try to interpret this message as a sequence of alphabets with a = 1, b = 2, c = 3, d = 4 and so on. Thus, one obvious interpretation of this message is as follows
[1] [2] [3] [4] => abcd

However, there exist other interpretations as well such as
[1][23][4] => awd
and
[12][3][4] => lcd

Note that [12][34] is not a valid interpretation since the English alphabet contains only 26 characters and there is no 34-th character. For the same reason, [123][4], [1][234], [1234] are also invalid. Now let us see what to do when there are zeros in the digits. Suppose n = 2 and the 2 digits are 10. Then one valid interpretation is
[10] => j
However, the interpretation [1][0] is not valid since there is no 0-th character in the English alphabet. Thus, the only interpretations that are valid are those in which each chunk corresponds to a number between 1 and 26 (both included).

We assure you that the messages that will be generated will take no more than 99 characters. We assure you that n will be no more than 99 either.

Compulsory function usage in your code
This is a question that benefits from use of recursion. In your code, you should write a function in the following format. Be warned that not using such a function to write your code will cause you to lose a small number of manual grading marks.

void decode(char *str, char *msg, int n, int done, int pos)

1. str: the string given to you as input
2. msg: the partially interpreted string
3. n: how many digits are there in str
4. done: how many digits of str have we processed yet
5. pos: which is the next position in the msg string where a character is to be inserted.

Function invocation: invoke the function as decode(str, msg, n, 0, 0); from the main function
Base case: the base case happens when we have done = n. In that case we can simply set msg[pos] = '\0' and print the string.

Recursive case: Think carefully about how you will ensure lexicographic order. Remember, you can either process the next digit in str (as indicated by done) as an English alphabet letter or you can process the next two digits in str (as indicated by done) as an English alphabet letter.

Problem-specific Words of Caution:

1. **Do not forget to submit your code**. You can submit multiple times. Your last submission will get graded.
2. We assure you that there will exist at least one interpretation. There may exist several as well in which case you have to output all of them in lexicographic order.
3. We will not penalize you for extra newlines at the end of your output. However, do not have stray spaces anywhere in your output.

General Grading Policy

1. **TOTAL MARKS OF THE EXAM** 20 + 40 + 40 + 70 = 170
2. **TOTAL DURATION OF THE EXAM** 3 hours 30 minutes
3. See below for question-specific details of how partial marking would be done by the autograder in this question
4. Your submissions will be inspected by the autograder as well as a human grader
5. Human graders will (among other things) allot marks for the following

    1. Neatly structured code that uses at least one function other than the main function to process the input. The questions will usually suggest how to use functions to process the input. Submissions that ignore these suggestions and use only the main function to solve the entire problem, will lose a small fraction of marks.

    2. Proper and meaningful variable names

    3. Nice looking and consistent indentation

    4. At least a couple of comments explaining to the human grader what are you doing, especially when the steps are not obvious

    5. Comments, good indentation and meaningful variable names are very important for the human grader to understand what are you doing and why. If they cannot understand your code, do not expect them to give you (partial) marks either.

6. Solutions that indulge in hard-coding **will get a straight zero** even if they are passing some test cases. Hard-coding is a form of cheating strategy where someone write code of the form "if(input == A) printf(B)" without doing any calculations on A to obtain B. The values of A and B are either read from the evaluation/submission window or else guessed.
7. Be careful about extra/missing lines and extra/missing spaces if you do not want to lose autograder marks
8. Proportion of marks allotted to autograder (in particular, weightage to visible and hidden test cases) and human grader will be revealed when marks and grading rubrics are released

9. You are allowed to use the libraries stdio.h, math.h, string.h, stdlib.h **but not any other library**. Use of unpermitted libraries will carry a penalty. You may use any programming tools that we have discussed in lectures/tutorials or in lab questions such as arrays (1D, 2D, 3D, arrays of arrays etc), strings, loops, structures, functions, recursion, pointers, linked lists, stacks, queues, graphs, enumerations, flags, conditionals, global, static and shadowed variables.

EXAMPLE:
INPUT
4
1234

OUTPUT:
abcd
awd
lcd

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. There are 4 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/6247)