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
  - Super Getline
  - Tailor-made Tabs
  - De-Duplicate
  - Matrix Mania
- 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

Super Getline

PRACTICE-10_MAT-FUN

The getline function from string.h automatically expands its length to incorporate additional characters entered by the user. However, it can only read a single line of the input at a time. I want to write a routine that can read the entire input, not just a single line (i.e. till EOF) and also be able to expand the array size as needed.

Start from the code given which begins with a character array of length just one and reads the entire user input (including newlines and other non-printable characters), all the while expanding the array in case more and more characters are found in the input. Print the entire user input .

Request: It is very easy to cheat in this practice question and pass all the test cases by repeating the input (character by character) as shown below. However if you do so, you will lose an opportunity to learn about pointers and memory allocation and this may affect you in exams where you will not be able to cheat this way.

Example of cheating:

```
do{
   printf("%c", ch = getchar());
}while(ch != EOF)
```

Bonus: write a function superGetline which takes as two arguments, the address of the original string array (in other words, a pointer to a pointer) and the address of an integer variable storing the length of the original array and does the above.

You should be able to solve this problem nicely even without using functions. Functions will be covered in class next week. Functions allow you to write cleaner code and allow you to think about the problem in an organized manner. However, although it is perfectly possible to solve most problems in a reasonable manner without using functions at all, code (especially if it is long and complex) written without functions, tends to be more error prone and hard to debug.

Start Solving! (/editor/practice/6178)