1873 - I2P(I)2019_Yang_CS_practice_Final

Scoreboard (/contest/scoreboard/1873/)

Time		
2019/12/17 21:00:00	12days, 18:58:55	2020/01/07 18:00:00

Clarification

#	Problem	Asker	Description	Reply	Replier	Reply Time	For all team

Overview | Problem

12094 - Coding in terminal

Status (/status/?pid=12094) | Limits | Submit (/users/submit/12094)

Description

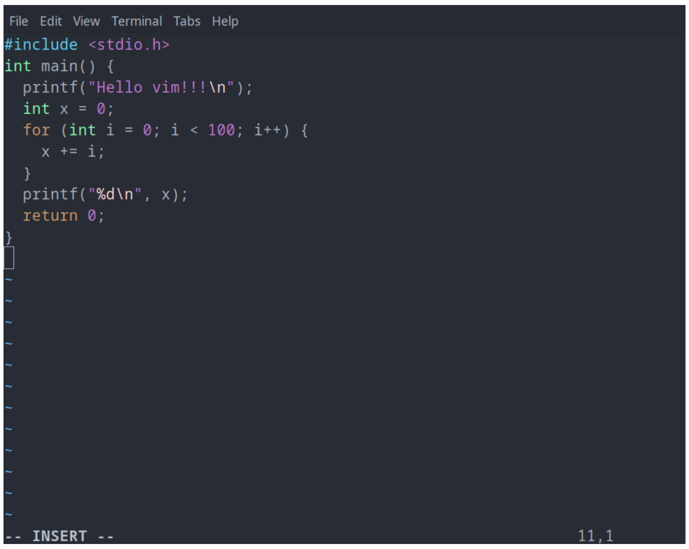
[Update] There's new-line characters ('\n') in the input test cases, you should ignore it.
[Update] You do not need to add an extra '\n' at the end of output!
[Notice] **Input contains space character (' ')**!

Some people prefer to use vim to code.

Vim is a powerful text editor. Using vim, you can code in a terminal. Usually we code with vim on a remote server, since there is no GUI applications (like CodeBlocks or Visual Studio Code) available.

Besides the basic functionality, you can install a variety of plugins to make vim looks fancy and convenient.

This is how it looks like to coding with vim



```
#include <stdio.h>
int main() {
  printf("Hello vim!!!\n");
  int x = 0;
  for (int i = 0; i < 100; i++) {
    x += i;
  }
  printf("%d\n", x);
  return 0;
}

~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                          11,1
```

Now, we ask you to simulate the most basic functionality of a text editor: typing.

Given a operation sequence, please show us the final look on the terminal. (Suppose we started in "Insert" state in vim, that is, what we type will be treated as text and directly shows on the screen. For those who don't know the "Insert" state, try google for it or just ignore it and keeps reading the problem description~)

Input

Input consists of a single line, representing the operation sequence.

Below we explain the meaning of each character in the operation sequence:

- Normal latin characters (uppercase/lowercase English characters) are treated as character input. Whatever the character is, show it on the screen.
- The special operations starts with a "\" (without quotes), follow by some operation indication characters. Below we list out all possible special operations:
  - "`\n`", user types Enter. Print a '\n' (new line character).
  - "`\b`", user types a backspace. Remove the character right before the cursor(游標) (if there is any).
  - "`\l`", move cursor left. If the cursor is at the beginning of current line, then go to the end of previous line (if exists).
  - "`\r`", move cursor right. If the cursor is at the end of current line, then go to the beginning of the next line (if exists).
  - "`\s [x y]`", change the terminal size. x and y are positive integers. There is only one space character ' ' between `s` and `[`, `x` and `y`.

Suppose the initial terminal size is 80*24. (**x**=80, **y**=24). **x** is the width of the terminal, means there can be **at most x characters** (including ' ') in a single line on the screen.

The final output is guaranteed to not exceed the size of the terminal. (Total lines on screen will be less than or equal to **y**)

The length of operation sequence won't exceed 10^5.

Output

Show the final look on the terminal.

Notice the width of the screen. If a **line** contains too many characters to fit into the width **x**, then you should print the remaining characters of this **line** on the following lines on screen.

For more information, please refer to Sample Input and Sample Output.

Sample Input

Download (data:text/plain;charset=utf-8,t%5Cb%5CbThis%20is%20the%20sample%20input%5Cl%5Cl%5Cl%5Cl%5CbI%5Cr%5Cr%5Cr%5Cr%5Cn%5Cs%20%5B10%2020%5DNotice%20the%20

```
t\b\bThis is the sample input\l\l\l\l\bI\r\r\r\r\n\s [10 20]Notice the \s [15 100]wideth\l\l\b\r\r of terminal
```

Sample Output

Download (data:text/plain;charset=utf-8,This%20is%20the%20sam%0Aple%20Input%0ANotice%20the%20widt%0Ah%20of%20terminal)

```
This is the sam
ple Input
Notice the widt
h of terminal
```

Discuss