实验 2-1: 简单的行编辑器

1. 问题描述

编写简单的行编辑器，完成对文本的编辑，主要实现以下 7 种操作：

1.1 `Move k`: 将光标移动到第k个字符之后，如果k=0则将光标移到文本第一个字符之前。

1.2 `Insert n S`: 在光标后插入长度为n的字符串S，光标位置不变，n≥1。

1.3 `Delete n`: 删除光标后的n个字符，光标位置不变，n≥1。

1.4 `Rotate n`: 反转光标后的n个字符，光标位置不变，n≥1。

1.5 `Get`: 输出此时程序内保存的文本

1.6 `Prev`: 光标前移一个字符。

1.7 `Next`: 光标后移一个字符。

输入要求: 输入文件中第一行是指令条数N，以下是需要执行的N个操作。除了回车符之外，输入文件的所有字符的ASCII码都在闭区间[32, 126]内。且行尾没有空格。

输出要求: 依次对应输入文件中每条Get指令的输出，不得有任何多余的字符。

2. 算法的描述

2.1 数据结构的描述

本程序主要的数据结构是 栈(Stack) ， 栈的实现和函数操作分别被定义在 `stack.c` `stack.h` 中，主函数在 `main.c` 中。

`main()` 中，定义了两个栈 `leftStack` 和 `rightStack` 用以确定光标的位置，此外，定义了变量 `inputMode` 用以确定是否在等待输入，即上一次的输入是否为 `Insert n`. `commandLine` 确定了当前已输入的命令数，`numLines` 定义了命令的总数. `input` 是一个数组，存储了输入的命令，由于每个命令的首字母不同，我们只需确定首字母即可确定命令。

2.2 程序结构的描述

`main.c` 中主要的函数有:

- `main()` 程序的入口，处理输入的各种命令。
- `PrintStack()` 输出整个链栈
- `PrintRStack()` 以倒序输出栈中每个元素
- `GetNum()` 处理输入命令中的数字 (n)

3. 调试分析

本程序使用的测试数据是助教提供的测试样例，在调试中出现了空间大小的问题，改用 malloc() 解决了问题

4. 算法的时间分析

上述操作的复杂度均为 O(N)