华为OD机试 - 代码编辑器（Java & JS & Python）

题目描述

某公司为了更高效的编写代码，邀请你开发一款代码编辑器程序。

程序的输入为 已有的代码文本和指令序列，程序需输出编辑后的最终文本。指针初始位置位于文本的开头。
支持的指令(X为大于等于0的整数, word 为无空格的字符串)：

- FORWARD X 指针向前(右)移动X,如果指针移动位置超过了文本末尾，则将指针移动到文本末尾
- BACKWARD X 指针向后(左)移动X,如果指针移动位置超过了文本开头，则将指针移动到文本开头
- SEARCH-FORWARD word 从指针当前位置向前查找 word 并将指针移动到word的起始位置，如果未找到则保持不变
- SEARCH-BACKWARD word 在文本中向后查我 word 并将指针移动到word的起始位置，如果未找到则保持不变
- INSERT word 在指针当前位置前插入word，并将指针移动到word的结尾
- REPLACE word 在指针当前位置替换并插入字符(删除原有字符，并增加新的字符)
- DELETE X 在指针位置删除X个字符

输入描述

输入的第一行为命令列表的长度K

输入的第二行为文件中的原始文本

接下来的K行，每行为一个指令

输出描述

编辑后的最终结果

备注

文本最长长度不超过 256K

用例

输入	1 ello INSERT h
输出	hello
说明	在文本开头插入

输入	2 hllo FORWARD 1 INSERT e
输出	hello
说明	在文本的第一个位置插入

输入	2 hell FORWARD 1000 INSERT o
输出	hello
说明	在文本的结尾插入

输入	1 hello REPLACE HELLO
输出	HELLO
说明	替换

输入	1 hello REPLACE HELLO_WORLD
输出	HELLO_WORLD
说明	超过文本长度替换

输入	2 hell FORWARD 10000 REPLACE O
输出	hellO
说明	超出文本长度替换

题目解析

本题考察逻辑分析，以及字符串常用操作。

本题题目描述对逻辑描述的非常清晰，大家可以对照题目描述来看下面的代码。

其中，容易出错的点是：

- SEARCH-BACKWARD 是向后查找，这个“向后”，其实是“向左”
- SEARCH-FORWARD 是向前查找，这个“向前”，其实是“向右”
- 假设指针位置是curIdx，原始文本是s，SEARCH-BACKWARD word，即相当于在s的0~curIdx范围反向查找word，这里容易出错的点是反向查找方法参数的含义，大家可以看下各自语言的下面方法参数的含义

Java	lastIndexOf	StringBuilder (Java Platform SE 8) (oracle.com)
JS	lastIndexOf	String.prototype.lastIndexOf() - JavaScript | MDN (mozilla.org)
Python	rfind	Python rfind()方法 | 菜鸟教程 (runoob.com)

- INSERT word，这里容易出错的点是，插入word到s中后，curIdx也要更新位置到插入word的最后一个单词位置后面

```java
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int k = Integer.parseInt(sc.nextLine());

    StringBuilder s = new StringBuilder(sc.nextLine());

    String[][] commands = new String[k][2];
    for (int i = 0; i < k; i++) {
      commands[i] = sc.nextLine().split(" ");
    }

    System.out.println(getResult(s, commands));
  }

  public static String getResult(StringBuilder s, String[][] commands) {
    int curIdx = 0;

    for (String[] command : commands) {
```

```
      switch (command[0]) {
        case "FORWARD":
          curIdx += Integer.parseInt(command[1]);
          curIdx = Math.min(curIdx, s.length());
          break;
        case "BACKWARD":
          curIdx -= Integer.parseInt(command[1]);
          curIdx = Math.max(curIdx, 0);
          break;
        case "SEARCH-FORWARD":
          int i = s.indexOf(command[1], curIdx);
          if (i != -1) curIdx = i;
          break;
        case "SEARCH-BACKWARD":
          int i1 = s.lastIndexOf(command[1], curIdx);
          if (i1 != -1) curIdx = i1;
          break;
        case "INSERT":
          s.insert(curIdx, command[1]);
          curIdx += command[1].length();
          break;
        case "REPLACE":
          s.replace(curIdx, curIdx + command[1].length(), command[1]);
          break;
        case "DELETE":
          s.delete(curIdx, curIdx + Integer.parseInt(command[1]));
          break;
      }
    }

    return s.toString();
  }
}
```