题目描述

九宫格按键输入，输出显示内容，有英文和数字两个模式，默认是数字模式，数字模式直接输出数字，英文模式连续按同一个按键会依次出现这个按键上的字母，如果输入"/"或者其他字符，则循环中断。

字符对应关系如图：

1 ,.	2 abc	3 def
4 ghi	5 jkl	6 mno
7 pqrs	8 tuv	9 wxyz
# 	0 空格	/

要求输入一串按键，输出屏幕显示。

输入描述

输入范围为数字 0~9 和字符'#'、'/'，输出屏幕显示，例如，

在数字模式下，输入 1234，显示 1234

在英文模式下，输入 1234，显示,adg

输出描述

#用于切换模式，默认是数字模式，执行#后切换为英文模式；

/表示延迟，例如在英文模式下，输入 22/222，显示为 bc；

英文模式下，多次按同一键，例如输入 22222，显示为 b；

用例

输入	123#222235/56
输出	123adjjm

题目解析

本题主要考察逻辑分析，和栈结构使用。

我的解题思路如下：

首先，定义一个栈stack，用于缓存结果。再定义一个isEng标识来记录当前模式，true代表英文模式，false代表数字模式。isEng初始化为false。

然后，遍历输入的字符串s，将s的每一个字符c遍历出来：

- 如果 c == '#'，则表示要切换模式，即让 isEng = !isEng，但是在切换模式之前，需要先检查isEng是否为英文模式，如果是英文模式，则切换前，需要先将栈顶数字的循环中断，并转化为对应的字母。
- 如果 c == '/'，则表示要循环中断，此时先检查isEng是否为英文模式，若是，则再将栈顶数字转化为对应字母，若不是，则不做处理
- 如果 c 为其他字符，则：

1. 如果isEng为数字模式，则c直接压入stack中
2. 如果isEng为英文模式，则需要检查stack栈顶元素是否和c相同，若不同，则需要循环中断，即将栈顶数字转化为字母，若相同，则栈顶数字重复次数++

本题循环中断的逻辑，需要做好预检查，即

- 当前必须是英文模式
- 栈不能为空

Java算法源码

```java
import java.util.LinkedList;
import java.util.Scanner;

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

  static String[] dict = {" ", ",.", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};

  static LinkedList<Character> stack = new LinkedList<>();
  static int topRepeat = 0;
  static boolean isEng = false;

  public static String getResult(String s) {
    s += " ";

    for (int i = 0; i < s.length(); i++) {
      char c = s.charAt(i);

```

```
22        switch (c) {
23          case '#':
24            // 如果输入"/"或者其他字符，则循环中断
25            interrupt();
26            // #用于切换模式
27            isEng = !isEng;
28            break;
29          case '/':
30            // 如果输入"/"或者其他字符，则循环中断
31            interrupt();
32            break;
33          default:
34            // 数字模式直接输出数字
35            if (!isEng) {
36              stack.add(c);
37              break;
38            }
39
40            // 英文模式下，如果栈为空，则缓存对应字符c，并记录重复次数
41            if (stack.size() == 0) {
42              stack.add(c);
43              topRepeat++;
44              break;
45            }
46
47            // 英文模式下，如果栈不为空
48            if (c != stack.getLast()) {
49              // 如果输入"/"或者其他字符，则循环中断
50              interrupt();
51              stack.add(c);
52            }
53            topRepeat++;
54        }
55      }
56
57      StringBuilder sb = new StringBuilder();
58      for (int i = 0; i < stack.size() - 1; i++) sb.append(stack.get(i));
59      return sb.toString();
60    }
61
62    // 英文模式连续按同一个按键会依次出现这个按键上的字母，如果输入"/"或者其他字符，则循环中断
63    // interrupt用于处理循环中断后的逻辑
64    public static void interrupt() {
65      if (!isEng || stack.size() == 0 || topRepeat == 0) return;
66      stack.add(map(stack.removeLast(), topRepeat));
67      topRepeat = 0;
68    }
69
70    // 基于dict，获取一个数字c按重复repeat次后，对应的字符
71    public static char map(char c, int repeat) {
72      int num = Integer.parseInt(c + "");
73      String s = dict[num];
74      int i = (repeat - 1) % s.length();
75      return s.charAt(i);
76    }
77  }
```

JS算法源码

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const stack = [];
let topRepeat = 0;
let isEng = false;

rl.on("line", (line) => {
  console.log(getResult(line));

  // 如果需要测试多组用例，这里需要将全局变量置
  stack.length = 0;
  topRepeat = 0;
  isEng = false;
});

function getResult(s) {
  s += " ";
  for (let c of s) {
    switch (c) {
      case "#":
        // 如果输入"/"或者其他字符，则循环中断
        interrupt();
        // #用于切换模式
        isEng = !isEng;
        break;
      case "/":
        // 如果输入"/"或者其他字符，则循环中断
        interrupt();
        break;
      default:
        // 数字模式直接输出数字
        if (!isEng) {
          stack.push(c);
          break;
        }

```

```
        // 英文模式下，如果栈为空，则缓存对应字符c，并记录重复次数
        if (stack.length == 0) {
          stack.push(c);
          topRepeat++;
          break;
        }

        // 英文模式下，如果栈不为空
        if (c != stack.at(-1)) {
          // 如果输入"/"或者其他字符，则循环中断
          interrupt();
          stack.push(c);
        }
        topRepeat++;
    }
  }

  return stack.slice(0, stack.length - 1).join("");
}

// 英文模式连续按同一个按键会依次出现这个按键上的字母，如果输入"/"或者其他字符，则循环中断
// interrupt用于处理循环中断后的逻辑
function interrupt() {
  if (!isEng || stack.length == 0 || topRepeat == 0) return;
  stack.push(map(stack.pop(), topRepeat));
  topRepeat = 0;
}

const dict = [
  " ",
  ",.",
  "abc",
  "def",
  "ghi",
  "jkl",
  "mno",
  "pqrs",
  "tuv",
  "wxyz",
];

// 基于dict，获取一个数字c被重复repeat次后，对应的字符
function map(c, repeat) {
  const num = parseInt(c);
  const s = dict[num];
  const i = (repeat - 1) % s.length;
  return s[i];
}
```

Python算法源码

```python
# 输入获取
s = input()

# 全局变量
stack = []
topRepeat = 0
isEng = False

dictionary = (" ", ",.", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz")


# 基于dictionary，获取一个数字c按重复repeat次后，对应的字符
def mapping(c, repeat):
    num = int(c)
    s1 = dictionary[num]
    i = (repeat - 1) % len(s1)
    return s1[i]


# 英文模式连续按同一个按键会依次出现这个按键上的字母，如果输入"/"或者其他字符，则循环中断
# interrupt用于处理循环中断后的逻辑
def interrupt():
    global topRepeat
    if not isEng or len(stack) == 0 or topRepeat == 0:
        return
    stack.append(mapping(stack.pop(), topRepeat))
    topRepeat = 0


# 算法入口
def getResult():
    global s
    global isEng
    global topRepeat

    s += " "

```

```
    for c in s:
        if c == '#':
            # 如果输入"/"或者其他字符，则循环中断
            interrupt()
            # #用于切换模式
            isEng = not isEng
        elif c == '/':
            # 如果输入"/"或者其他字符，则循环中断
            interrupt()
        else:
            # 数字模式直接输出数字
            if not isEng:
                stack.append(c)
                continue

            # 英文模式下，如果栈为空，则缓存对应字符c，并记录重复次数
            if len(stack) == 0:
                stack.append(c)
                topRepeat += 1
                continue

            # 英文模式下，如果栈不为空
            if c != stack[-1]:
                # 如果输入"/"或者其他字符，则循环中断
                interrupt()
                stack.append(c)

            topRepeat += 1

    return "".join(stack[:-1])


# 算法调用
print(getResult())
```