题目描述

已知火星人使用的运算符为#、$，其与地球人的等价公式如下：

x#y = 2*x+3*y+4

x$y = 3*x+y+2

1. 其中x、y是无符号整数
2. 地球人公式按C语言规则计算
3. 火星人公式中，$的优先级高于#，相同的运算符，按从左到右的顺序计算

现有一段火星人的字符串报文，请你来翻译并计算结果。

输入描述

火星人字符串表达式（结尾不带回车换行）

输入的字符串说明：字符串为仅由无符号整数和操作符（#、$）组成的计算表达式。

例如：123#4$5#67$78。

1. 用例保证字符串中，操作数与操作符之间没有任何分隔符。
2. 用例保证操作数取值范围为32位无符号整数。
3. 保证输入以及计算结果不会出现整型溢出。
4. 保证输入的字符串为合法的求值报文，例如：123#4$5#67$78
5. 保证不会出现非法的求值报文，例如类似这样字符串：

#4$5 //缺少操作数

4$5# //缺少操作数

4#$5 //缺少操作数

4 $5 //有空格

3+4-5*6/7 //有其它操作符

123456789876543215$54321 //32位整数计算溢出

输出描述

根据输入的火星人字符串输出计算结果（结尾不带回车换行）。

用例

输入	7#6$5#12
输出	226
说明	7#6$5#12 =7#(3*6+5+2)#12 =7#25#12 =(2*7+3*25+4)#12 =93#12 =2*93+3*12+4 =226

题目解析

这题一开始用栈结构来做，很难，因为$要优先计算，并且还要找到$两边的操作数，因此基于栈结构，就不好操作了。

后面，想了一下，这个题目保证不会出现非法的求值报文，因此输入字符串是严格的"数字+操作符+数字"这种格式，因此，很适合使用正则去匹配。

首先，我先用正则匹配出"操作数$操作数"，然后将其替换为计算后的值，然后字符串中就只剩#了，因此再将字符串按照#分割，从左到右，两两操作计算。

最终就得到了题解。

JavaScript算法源码

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

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

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

function huoxin(str) {
  const regExp = /\d+\$\d+/;
  while (str.indexOf("$") !== -1) {
    str = str.replace(regExp, (match) => operate$(match).toString());
  }

  return str
    .split("#")
    .map(Number)
    .reduce((x, y) => 2 * x + 3 * y + 4);
}

function operate$(str) {
  const i = str.indexOf("$");
  const x = parseInt(str.slice(0, i));
  const y = parseInt(str.slice(i + 1));
  return 3 * x + y + 2;
}
```

Java算法源码

```java
import java.util.Arrays;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

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

  public static int getResult(String str) {
    Pattern p = Pattern.compile("(\\d+)\\$(\\d+)");

    while (true) {
      Matcher m = p.matcher(str);
      if (!m.find()) break;

      String subStr = m.group(0);
      int x = Integer.parseInt(m.group(1));
      int y = Integer.parseInt(m.group(2));
      str = str.replace(subStr, 3 * x + y + 2 + "");
    }

    return Arrays.stream(str.split("#"))
        .map(Integer::parseInt)
        .reduce((x, y) -> 2 * x + 3 * y + 4)
        .orElse(0);
  }
}
```

Python算法源码

```python
import re

# 输入获取
s = input()


# 算法入口
def getResult(s):
    p = re.compile("(\\d+)\\$(\\d+)")

    while True:
        m = p.search(s)
        if m:
            subS = m.group()
            x = int(m.group(1))
            y = int(m.group(2))
            s = s.replace(subS, str(3 * x + y + 2))
        else:
            break

    arr = list(map(int, s.split("#")))

    x = arr[0]
    for y in arr[1:]:
        x = 2 * x + 3 * y + 4

    return x


# 算法调用
print(getResult(s))
```