题目描述

你现在是一场采用特殊赛制投篮大赛的记录员。这场比赛由若干回合组成，过去几回合的得分可能会影响以后几回合的得分。
比赛开始时，记录是空白的。
你会得到一个记录操作的字符串列表 ops，其中ops[i]是你需要记录的第i项操作，ops遵循下述规则：

- 整数x-表示本回合新获得分数x
- “+” – 表示本回合新获得的得分是前两次得分的总和。
- “D” – 表示本回合新获得的得分是前一次得分的两倍。
- “C” – 表示本回合没有分数，并且前一次得分无效，将其从记录中移除。

请你返回记录中所有得分的总和。

输入描述

输入为一个 字符串数组

输出描述

输出为一个整形数字

提示

1. 1 <= ops.length <= 1000
2. ops[i] 为 “C”、“D”、“+”，或者一个表示整数的字符串。整数范围是 [-3 * 10^4, 3 * 10^4]
3. 需要考虑异常的存在，如有异常情况，请返回-1
4. 对于“+”操作，题目数据不保证记录此操作时前面总是存在两个有效的分数
5. 对于“C”和“D”操作，题目数据不保证记录此操作时前面存在一个有效的分数
6. 题目输出范围不会超过整型的最大范围，不超过2^63 - 1

用例

输入	5 2 C D +
输出	30
说明	“5”-记录加5，记录现在是[5] “2”-记录加2，记录现在是[5,2] “C”-使前一次得分的记录无效并将其移除，记录现在是[5]. “D”-记录加2*5=10，记录现在是[5，10]. “+”-记录加5+10=15，记录现在是[5，10，15]. 所有得分的总和5+10+15=30

输入	5 -2 4 C D 9 + +
输出	27
说明	“5”-记录加5，记录现在是[5] “-2”-记录加-2，记录现在是[5,-2] “4”-记录加4，记录现在是[5,-2,4] “C”-使前一次得分的记录无效并将其移除，记录现在是[5,-2]. “D”-记录加2*-2=4，记录现在是[5，-2, -4]. “9”-记录加9，记录现在是[5，-2, -4, 9]. “+”-记录加-4+9=5，记录现在是[5，-2, -4, 9, 5]. “+”-记录加-9+5=14，记录现在是[5，-2, -4, 9, 5, 14]. 所以得分的总和 5 - 2 - 4 + 9 + 5 + 14 = 27

用例

输入	5 2 C D +
输出	30
说明	“5”-记录加5，记录现在是[5] “2”-记录加2，记录现在是[5,2] “C”-使前一次得分的记录无效并将其移除，记录现在是[5]. “D”-记录加2*5=10，记录现在是[5，10]. “+”-记录加5+10=15，记录现在是[5，10，15]. 所有得分的总和5+10+15=30

输入	5 -2 4 C D 9 + +
输出	27
说明	“5”-记录加5，记录现在是[5] “-2”-记录加-2，记录现在是[5,-2] “4”-记录加4，记录现在是[5,-2,4] “C”-使前一次得分的记录无效并将其移除，记录现在是[5,-2]. “D”-记录加2*-2=4，记录现在是[5，-2, -4]. “9”-记录加9，记录现在是[5，-2, -4, 9]. “+”-记录加-4+9=5，记录现在是[5，-2, -4, 9, 5]. “+”-记录加-9+5=14，记录现在是[5，-2, -4, 9, 5, 14]. 所以得分的总和 5 - 2 - 4 + 9 + 5 + 14 = 27

输入	1
输出	1
说明	无

输入	+
输出	-1
说明	无

题目解析

简单的逻辑题，按照题目意思写就行。

JavaScript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,
});

rl.on("line", (line) => {
  const ops = line.split(" ");
  console.log(getResult(ops));
});

function getResult(ops) {
  // ans用于保存每轮的得分
  const ans = [];

  const reg = /^\-?\d+$/;
  for (let op of ops) {
    // 如果op是整数，则表示本轮得分，直接加入ans
    if (reg.test(op)) {
      ans.push(op - 0);
    } else {
      switch (op) {
        // 如果op是+，则表示本轮得分是前两轮得分之和，注意越界处理
        case "+":
          if (!ans.at(-1) || !ans.at(-2)) return -1;
          ans.push(ans.at(-1) + ans.at(-2));
          break;
        // 如果op是D，表示本轮得分是前一轮得分的双倍，注意越界处理
        case "D":
          if (!ans.at(-1)) return -1;
```

题目解析

简单的逻辑题，按照题目意思写就行。

JavaScript算法源码

```javascript
/* 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,
});

rl.on("line", (line) => {
  const ops = line.split(" ");
  console.log(getResult(ops));
});

function getResult(ops) {
  // ans用于保存每轮的得分
  const ans = [];

  const reg = /^\-?\d+$/;
  for (let op of ops) {
    // 如果op是整数，则表示本轮得分，直接加入ans
    if (reg.test(op)) {
      ans.push(op - 0);
    } else {
      switch (op) {
        // 如果op是+，则表示本轮得分是前两轮得分之和，注意越界处理
        case "+":
          if (!ans.at(-1) || !ans.at(-2)) return -1;
          ans.push(ans.at(-1) + ans.at(-2));
          break;
        // 如果op是D，表示本轮得分是前一轮得分的双倍，注意越界处理
        case "D":
          if (!ans.at(-1)) return -1;
          ans.push(ans.at(-1) * 2);
          break;
        // 如果op是C，则表示本轮无得分，且上一轮得分无效，需要去除
        case "C":
          if (!ans.at(-1)) return -1;
          ans.pop();
          break;
      }
    }
  }

  // 感谢网友m0_71826536提示，如果用例输入为：2 C，则此处会报错
  // return ans.reduce((p, c) => p + c);

  if (ans.length) return ans.reduce((p, c) => p + c);
  else return 0;
}
```

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);

    String str = sc.nextLine();

    String[] ops = str.split(" ");

    System.out.println(getResult(ops));
  }

  public static int getResult(String[] ops) {
    // ans用于保存每轮的得分
    LinkedList<Integer> ans = new LinkedList<>();
    String reg = "^\\-?\\d+$";

    for (String op : ops) {
      // 如果op是整数，则表示本轮得分，直接加入ans
      if (op.matches(reg)) {
        ans.addLast(Integer.parseInt(op));
      } else {
        switch (op) {
            // 如果op是+，则表示本轮得分是前两轮得分之和，注意越界处理
          case "+":
            if (ans.size() < 2) return -1;
            ans.addLast(ans.getLast() + ans.get(ans.size() - 2));
            break;
            // 如果op是D，表示本轮得分是前一轮得分的双倍，注意越界处理
          case "D":
            if (ans.size() < 1) return -1;
            ans.addLast(ans.getLast() * 2);
```

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);

    String str = sc.nextLine();

    String[] ops = str.split(" ");

    System.out.println(getResult(ops));
  }

  public static int getResult(String[] ops) {
    // ans用于保存每轮的得分
    LinkedList<Integer> ans = new LinkedList<>();
    String reg = "^\\-?\\d+$";

    for (String op : ops) {
      // 如果op是整数，则表示本轮得分，直接加入ans
      if (op.matches(reg)) {
        ans.addLast(Integer.parseInt(op));
      } else {
        switch (op) {
            // 如果op是+，则表示本轮得分是前两轮得分之和，注意越界处理
          case "+":
            if (ans.size() < 2) return -1;
            ans.addLast(ans.getLast() + ans.get(ans.size() - 2));
            break;
            // 如果op是D，表示本轮得分是前一轮得分的双倍，注意越界处理
          case "D":
            if (ans.size() < 1) return -1;
            ans.addLast(ans.getLast() * 2);
            break;
            // 如果op是C，则表示本轮无得分，且上一轮得分无效，需要去除
          case "C":
            // 感谢网友m0_71826536的提示，由于题目说：对于“C”和“D”操作，题目数据不保证记录此操作时前面存在一个有效的分数，
            if (ans.size() < 1) return -1;
            ans.removeLast();
            break;
        }
      }
    }

    int sum = 0;
    for (Integer an : ans) {
      sum += an;
    }
    return sum;
  }
}
```

Python算法源码

```python
import re

# 输入获取
ops = input().split()


# 算法入口
def getResult(ops):
    # ans用于保存每轮的得分
    ans = []

    # 改正则用于判断op是否为一个整数，包括负数
    pattern = r"^\-?\d+$"

    for op in ops:
        # 如果op是整数，则表示本轮得分，直接加入ans
        if re.match(pattern, op):
            ans.append(int(op))
        else:
            # 如果op是+，则表示本轮得分是前两轮得分之和，注意越界处理
            if op == "+":
                if len(ans) < 2:
                    return -1
                else:
                    ans.append(ans[-1] + ans[-2])
            # 如果op是D，表示本轮得分是前一轮得分的双倍，注意越界处理
            elif op == "D":
                if len(ans) < 1:
                    return -1
                else:
                    ans.append(ans[-1] * 2)
            # 如果op是C，则表示本轮无得分，且上一轮得分无效，需要去除
            elif op == "C":
                if len(ans) < 1:
                    return -1
```

```
        switch (op) {
            // 如果op是+，则表示本轮得分是前两轮得分之和，注意越界处理
          case "+":
            if (ans.size() < 2) return -1;
            ans.addLast(ans.getLast() + ans.get(ans.size() - 2));
            break;
            // 如果op是D，表示本轮得分是前一轮得分的双倍，注意越界处理
          case "D":
            if (ans.size() < 1) return -1;
            ans.addLast(ans.getLast() * 2);
            break;
            // 如果op是C，则表示本轮无得分，且上一轮得分无效，需要去除
          case "C":
            // 感谢网友m0_71826536的提示，由于题目说：对于"C"和"D"操作，题目数据不保证记录此操作时前面存在一个有效的分数，
            if (ans.size() < 1) return -1;
            ans.removeLast();
            break;
        }
      }
    }

    int sum = 0;
    for (Integer an : ans) {
      sum += an;
    }
    return sum;
  }
}
```

Python算法源码

```
import re

# 输入获取
ops = input().split()


# 算法入口
def getResult(ops):
    # ans用于保存每轮的得分
    ans = []

    # 改正则用于判断op是否为一个整数，包括负数
    pattern = r"^\-?\d+$"

    for op in ops:
        # 如果op是整数，则表示本轮得分，直接加入ans
        if re.match(pattern, op):
            ans.append(int(op))
        else:
            # 如果op是+，则表示本轮得分是前两轮得分之和，注意越界处理
            if op == "+":
                if len(ans) < 2:
                    return -1
                else:
                    ans.append(ans[-1] + ans[-2])
            # 如果op是D，表示本轮得分是前一轮得分的双倍，注意越界处理
            elif op == "D":
                if len(ans) < 1:
                    return -1
                else:
                    ans.append(ans[-1] * 2)
            # 如果op是C，则表示本轮无得分，且上一轮得分无效，需要去除
            elif op == "C":
                if len(ans) < 1:
                    return -1
                else:
                    ans.pop()
            else:
                return -1

    ans.append(0)

    return sum(ans)


# 调用算法
print(getResult(ops))
```