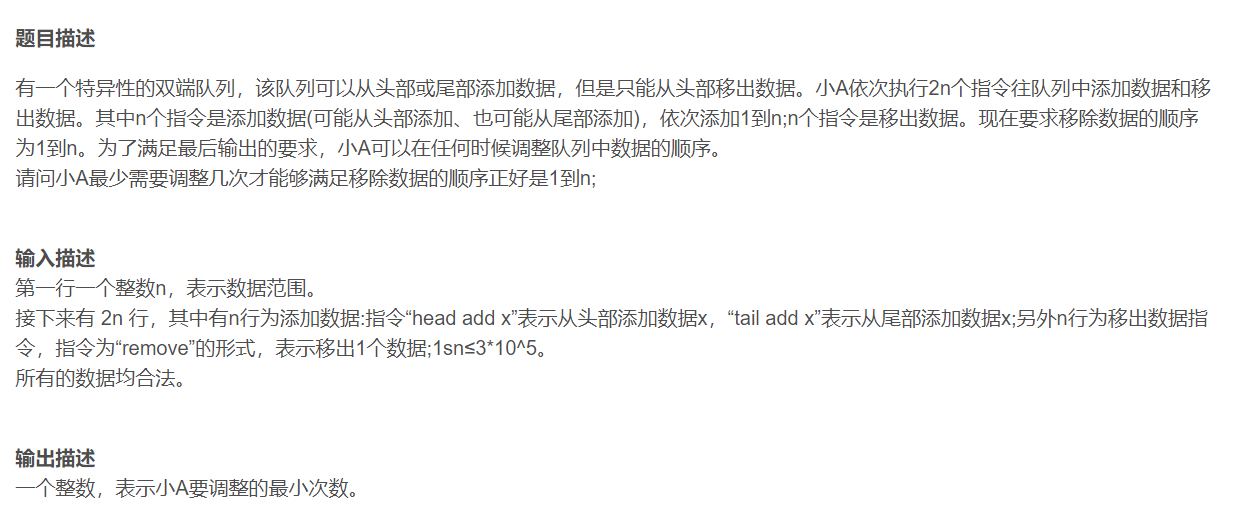
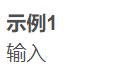
# **E卷-最小的调整次数[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**





3

head add 1

remove

tail add 2

head add 3

remove

Remove



1



输入命令 队列

head add 1 1

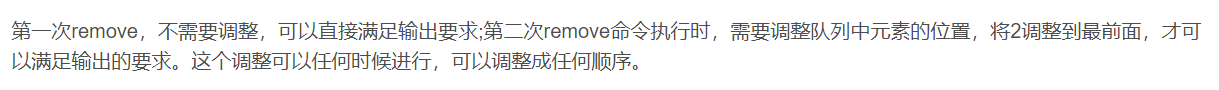
remove 空

tail add 2 2

head add 3 3 2

remove 3

remove 空





import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 读取输入的整数 n，表示数据范围

int n = scanner.nextInt();

scanner.nextLine(); // 清除行尾的换行符

// 初始化变量：

int isSorted =

1; // 当前队列中的数据是否按顺序排列（1表示是，0表示否）

int res = 0; // 需要调整的最小次数

int sz = 0; // 当前队列中的数据数量

// 遍历2n次操作指令

for (int i = 0; i < (n << 1); i++) {

String[] op = scanner.nextLine().split(" ");

// 如果指令是 "head add x"

if (op[0].equals("head")) {

if (sz >= 1) {

isSorted = 0;

}

sz += 1;

}

// 如果指令是 "tail add x"

else if (op[0].equals("tail")) {

sz += 1;

}

// 如果指令是 "remove"

else if (op[0].equals("remove")) {

if (isSorted == 0) {

isSorted = 1;

res += 1;

}

sz -= 1;

}

}

// 输出最终需要进行调整的最小次数

System.out.println(res);

}

}



# 读取输入的整数 n，表示数据范围。

n = int(input())

# 初始化变量：

# is\_sorted 表示当前队列中的数据是否按顺序排列。

# res 表示需要调整的最小次数。

# sz 表示当前队列中的数据数量。

is\_sorted = 1

res = 0

sz = 0

# 遍历 2n 次操作输入指令。

for i in range(n << 1): # n << 1 相当于 2n，表示共 2n 次操作。

# 读取并拆分当前操作指令。

op = input().split()

# 如果指令是 "head add x"：

if op[0] == "head":

# 如果当前队列中已经存在数据，则表示数据可能没有按顺序排列。

if sz >= 1:

is\_sorted = 0

# 队列长度增加。

sz += 1

# 如果指令是 "tail add x"：

if op[0] == "tail":

# 仅增加队列长度。

sz += 1

# 如果指令是 "remove"：

if op[0] == "remove":

# 如果当前队列中的数据没有按顺序排列，则需要进行调整。

if not is\_sorted:

is\_sorted = 1

res += 1

# 队列长度减少。

sz -= 1

# 输出最终需要进行调整的最小次数。

print(res)



#include <iostream>

#include <string>

using namespace std;

int main() {

int n;

// 读取输入的整数 n，表示数据范围

cin >> n;

cin.ignore(); // 清除行尾的换行符

// 初始化变量：

int is\_sorted =

1; // 当前队列中的数据是否按顺序排列（1表示是，0表示否）

int res = 0; // 需要调整的最小次数

int sz = 0; // 当前队列中的数据数量

// 遍历 2n 次操作指令

for (int i = 0; i < (n << 1); i++) {

string op;

getline(cin, op); // 读取一整行指令

// 如果指令是 "head add x"

if (op.substr(0, 4) == "head") {

if (sz >= 1) {

is\_sorted = 0;

}

sz += 1;

}

// 如果指令是 "tail add x"

else if (op.substr(0, 4) == "tail") {

sz += 1;

}

// 如果指令是 "remove"

else if (op == "remove") {

if (is\_sorted == 0) {

is\_sorted = 1;

res += 1;

}

sz -= 1;

}

}

// 输出最终需要进行调整的最小次数

cout << res << endl;

return 0;

}



#include <stdio.h>

#include <string.h>

int main() {

int n;

// 读取输入的整数 n，表示数据范围

scanf("%d", &n);

getchar(); // 清除行尾的换行符

// 初始化变量：

int is\_sorted =

1; // 当前队列中的数据是否按顺序排列（1表示是，0表示否）

int res = 0; // 需要调整的最小次数

int sz = 0; // 当前队列中的数据数量

char op[20]; // 用于存储操作指令

// 遍历 2n 次操作指令

for (int i = 0; i < (n << 1); i++) {

fgets(op, sizeof(op), stdin); // 读取一整行指令，并存到op数组中

// 如果指令是 "head add x"

if (strncmp(op, "head", 4) == 0) {

if (sz >= 1) {

is\_sorted = 0;

}

sz += 1;

}

// 如果指令是 "tail add x"

else if (strncmp(op, "tail", 4) == 0) {

sz += 1;

}

// 如果指令是 "remove"

else if (strncmp(op, "remove", 6) == 0) {

if (is\_sorted == 0) {

is\_sorted = 1;

res += 1;

}

sz -= 1;

}

}

// 输出最终需要进行调整的最小次数

printf("%d\n", res);

return 0;

}



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

let input = [];

let n;

rl.on("line", function (line) {

input.push(line);

}).on("close", function () {

// 读取输入的整数 n，表示数据范围

n = parseInt(input.shift());

// 初始化变量：

let isSorted = 1; // 当前队列中的数据是否按顺序排列（1表示是，0表示否）

let res = 0; // 需要调整的最小次数

let sz = 0; // 当前队列中的数据数量

// 遍历 2n 次操作指令

for (let i = 0; i < n << 1; i++) {

let op = input[i].split(" ");

// 如果指令是 "head add x"

if (op[0] === "head") {

if (sz >= 1) {

isSorted = 0;

}

sz += 1;

}

// 如果指令是 "tail add x"

else if (op[0] === "tail") {

sz += 1;

}

// 如果指令是 "remove"

else if (op[0] === "remove") {

if (!isSorted) {

isSorted = 1;

res += 1;

}

sz -= 1;

}

}

// 输出最终需要进行调整的最小次数

console.log(res);

});



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

func main() {

reader := bufio.NewReader(os.Stdin)

// 读取输入的整数 n，表示数据范围

nStr, \_ := reader.ReadString('\n')

n, \_ := strconv.Atoi(strings.TrimSpace(nStr))

// 初始化变量：

isSorted := 1 // 当前队列中的数据是否按顺序排列（1表示是，0表示否）

res := 0 // 需要调整的最小次数

sz := 0 // 当前队列中的数据数量

// 遍历 2n 次操作指令

for i := 0; i < (n << 1); i++ {

op, \_ := reader.ReadString('\n')

op = strings.TrimSpace(op)

// 如果指令是 "head add x"

if strings.HasPrefix(op, "head") {

if sz >= 1 {

isSorted = 0

}

sz += 1

}

// 如果指令是 "tail add x"

if strings.HasPrefix(op, "tail") {

sz += 1

}

// 如果指令是 "remove"

if op == "remove" {

if isSorted == 0 {

isSorted = 1

res += 1

}

sz -= 1

}

}

// 输出最终需要进行调整的最小次数

fmt.Println(res)

}