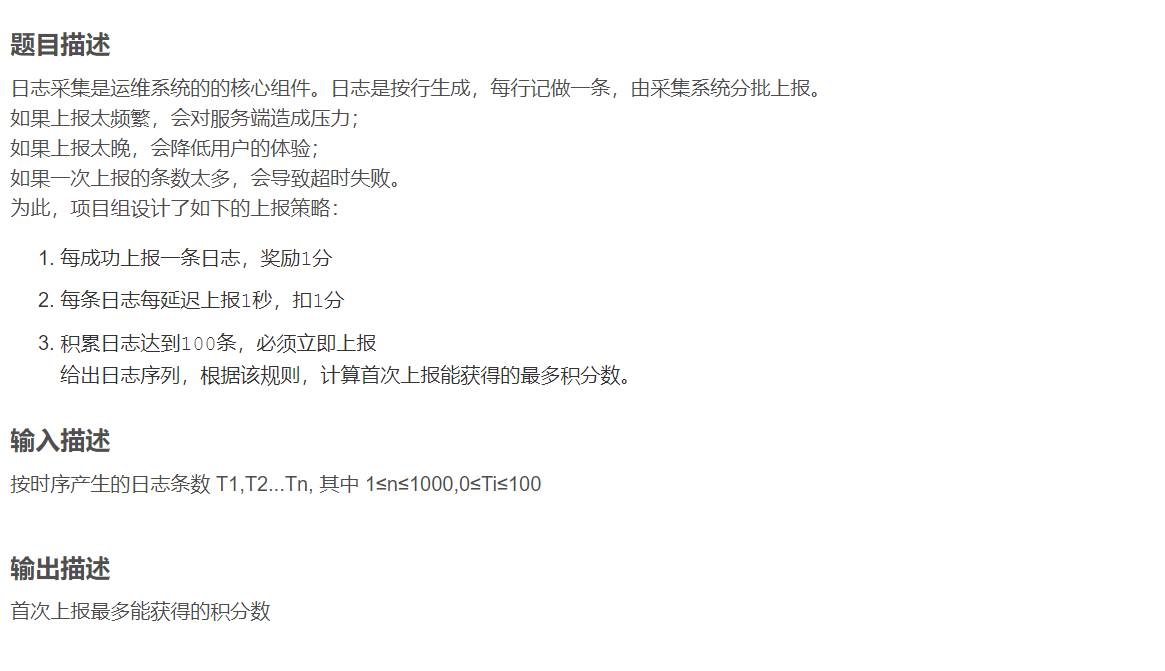
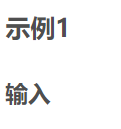
# **E卷-日志采集系统[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**

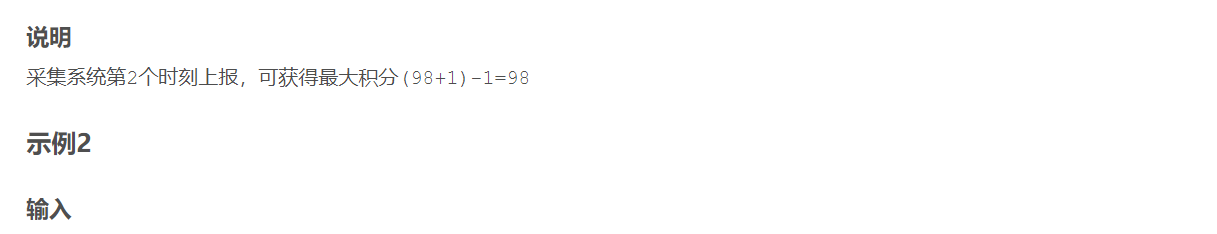




1 98 1



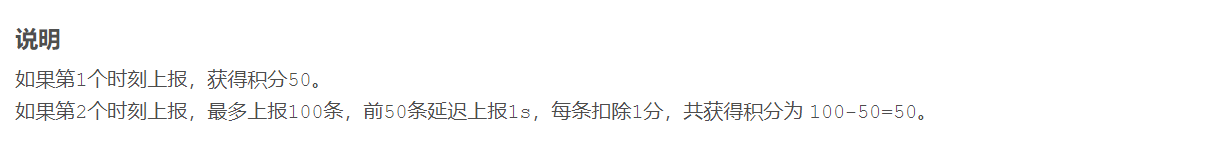
98



50 60 1



50





import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

List<Integer> arr = new ArrayList<>();

// 读取输入的日志条数

while (sc.hasNextInt()) {

arr.add(sc.nextInt());

}

// 当前的累计日志条数

int total = arr.get(0);

// 当前的最大积分数，初始值为首次读取到的日志条数或100，取较小值

int ans = Math.min(100, arr.get(0));

int sub = 0; // 初始化延迟扣分数

// 遍历日志条数列表，计算最大积分

for (int i = 1; i < arr.size(); i++) {

sub += total; // 累加当前累计的日志条数到sub

total += arr.get(i); // 累加新读取的日志条数到total

ans = Math.max(ans, Math.min(total, 100) - sub); // 更新最大积分数

// 当累计日志条数达到或超过100时，结束循环

if (total >= 100) {

break;

}

}

// 输出最大积分数

System.out.println(ans);

}

}



# 读取输入

arr = list(map(int, input().split()))

# 当前的累计日志条数

total = arr[0]

# 当前的最大积分数，初始值为首次读取到的日志条数或100，取较小值

ans = min(100, arr[0])

sub = 0 # 初始化延迟扣分数

# 遍历日志条数列表，计算最大积分

for i in range(1, len(arr)):

sub += total # 累加当前累计的日志条数到sub

total += arr[i] # 累加新读取的日志条数到total

ans = max(ans, min(total, 100) - sub) # 更新最大积分数

# 当累计日志条数达到或超过100时，结束循环

if total >= 100:

break

# 输出最大积分数

print(ans)



#include <bits/stdc++.h>

using namespace std;

int main() {

vector<int> arr; // 存储日志条数的数组

int x;

// 通过标准输入读取日志条数

// 直到输入结束，将所有读入的条数存入数组中

while (cin >> x) {

arr.push\_back(x);

}

// total表示当前累计的日志条数

int total = arr[0];

// ans表示当前最大积分，初始值为第一次读取到的日志条数（若日志条数超过100，则设置为100）

int ans = min(100, arr[0]);

int sub = 0; // sub表示累计的延迟扣分数

// 遍历读取到的日志条数数组

for (int i = 1; i < arr.size(); i++) {

sub += total; // 累加当前的日志条数到sub，表示累计的延迟

total += arr[i]; // 更新累计日志条数

ans = max(ans, min(total, 100) - sub); // 计算当前最大积分

if (total >= 100) {

// 当累计日志条数超过或等于100时，直接结束计算

break;

}

}

// 输出首次上报能获得的最大积分数

cout << ans << endl;

return 0;

}



#include <stdio.h>

int main() {

int arr[1000]; // 假设输入最多1000个整数

int n = 0;

// 读取输入的日志条数

while (scanf("%d", &arr[n]) != EOF) {

n++;

}

// 当前的累计日志条数

int total = arr[0];

// 当前的最大积分数，初始值为首次读取到的日志条数或100，取较小值

int ans = total < 100 ? total : 100;

int sub = 0; // 初始化延迟扣分数

// 遍历日志条数列表，计算最大积分

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

sub += total; // 累加当前累计的日志条数到sub

total += arr[i]; // 累加新读取的日志条数到total

if (total < 100) {

// 更新最大积分数

if (ans < total - sub) {

ans = total - sub;

}

} else {

if (ans < 100 - sub) {

ans = 100 - sub;

}

break; // 当累计日志条数达到或超过100时，结束循环

}

}

// 输出最大积分数

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

return 0;

}



const readline = require("readline");

// 创建接口读取输入

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

let arr = []; // 用于存储输入的日志条数

// 每行输入时执行的回调函数

rl.on("line", (line) => {

arr = line.split(" ").map(Number);

rl.close();

});

// 在接口关闭时执行的回调函数

rl.on("close", () => {

let total = arr[0];

let ans = Math.min(100, arr[0]); // 初始最大积分数

let sub = 0; // 初始化延迟扣分数

// 遍历日志条数列表，计算最大积分

for (let i = 1; i < arr.length; i++) {

sub += total; // 累加当前累计的日志条数到sub

total += arr[i]; // 累加新读取的日志条数到total

ans = Math.max(ans, Math.min(total, 100) - sub); // 更新最大积分数

// 当累计日志条数达到或超过100时，结束循环

if (total >= 100) {

break;

}

}

// 输出最大积分数

console.log(ans);

});



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

func main() {

reader := bufio.NewReader(os.Stdin)

line, \_ := reader.ReadString('\n') // 读取输入的一行数据

items := strings.Fields(line) // 按空格拆分字符串

arr := make([]int, len(items))

for i, s := range items {

arr[i], \_ = strconv.Atoi(s) // 转换为整数并存储在数组中

}

// 当前的累计日志条数

total := arr[0]

// 当前的最大积分数，初始值为首次读取到的日志条数或100，取较小值

ans := min(100, arr[0])

sub := 0 // 初始化延迟扣分数

// 遍历日志条数列表，计算最大积分

for i := 1; i < len(arr); i++ {

sub += total // 累加当前累计的日志条数到sub

total += arr[i] // 累加新读取的日志条数到total

ans = max(ans, min(total, 100)-sub) // 更新最大积分数

// 当累计日志条数达到或超过100时，结束循环

if total >= 100 {

break

}

}

// 输出最大积分数

fmt.Println(ans)

}

// 返回两个整数中的最小值

func min(a, b int) int {

if a < b {

return a

}

return b

}

// 返回两个整数中的最大值

func max(a, b int) int {

if a > b {

return a

}

return b

}