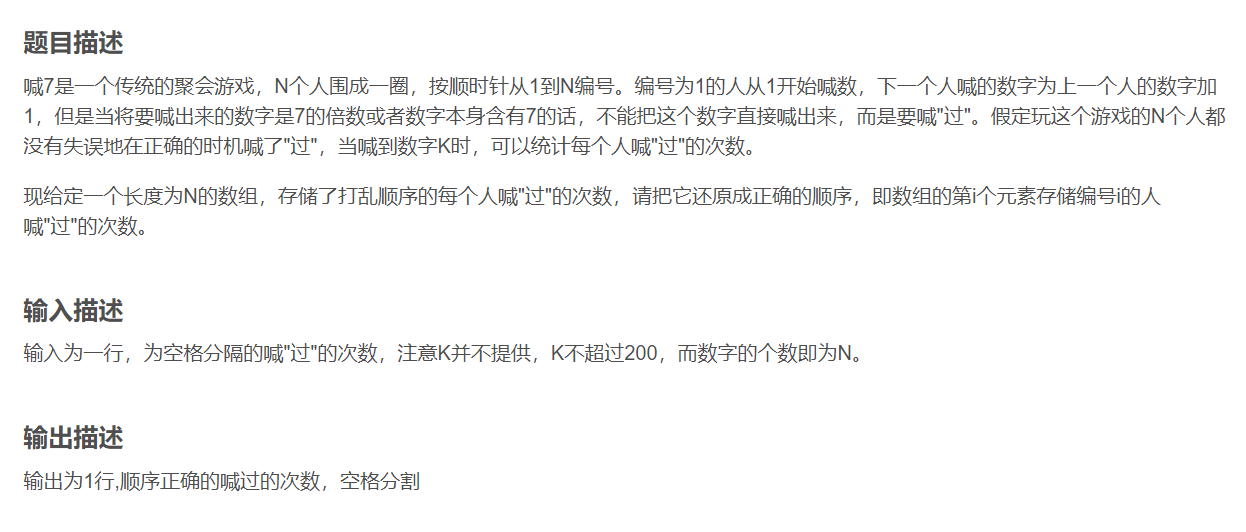
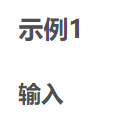
# **E卷-喊7的次数重排[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**

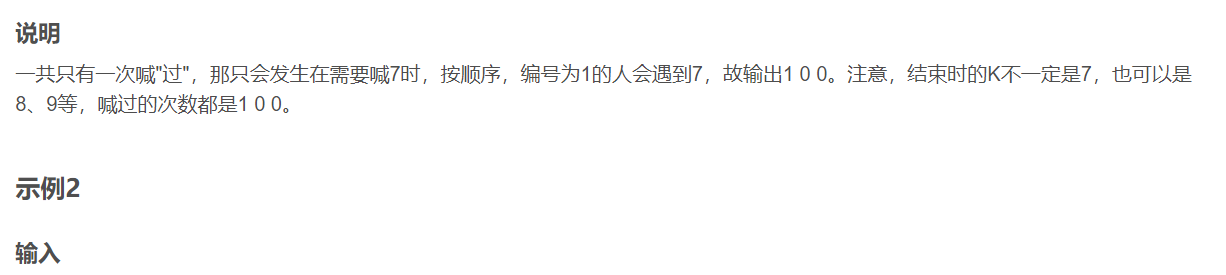




0 1 0



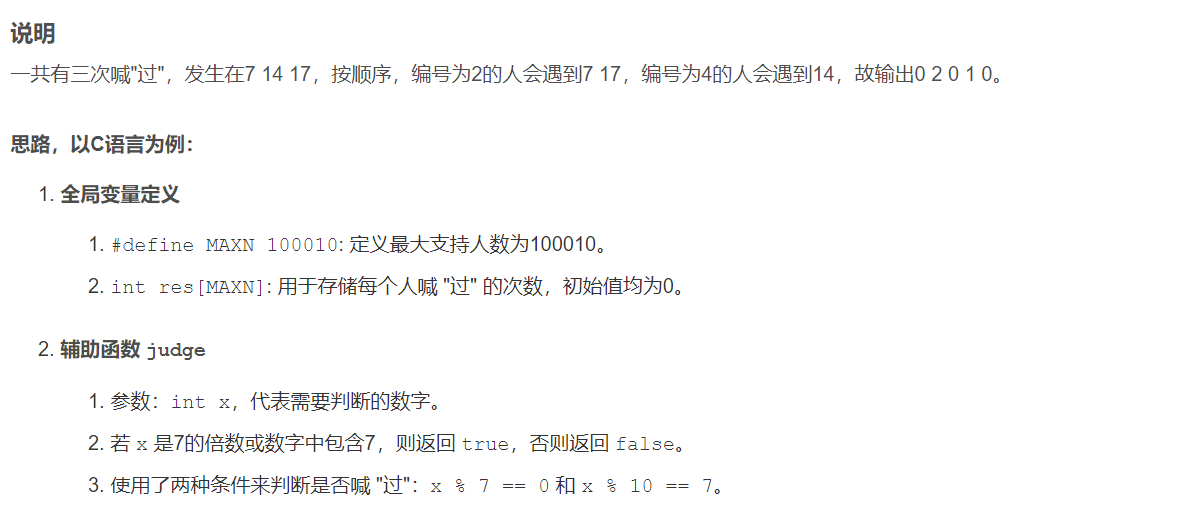
1 0 0

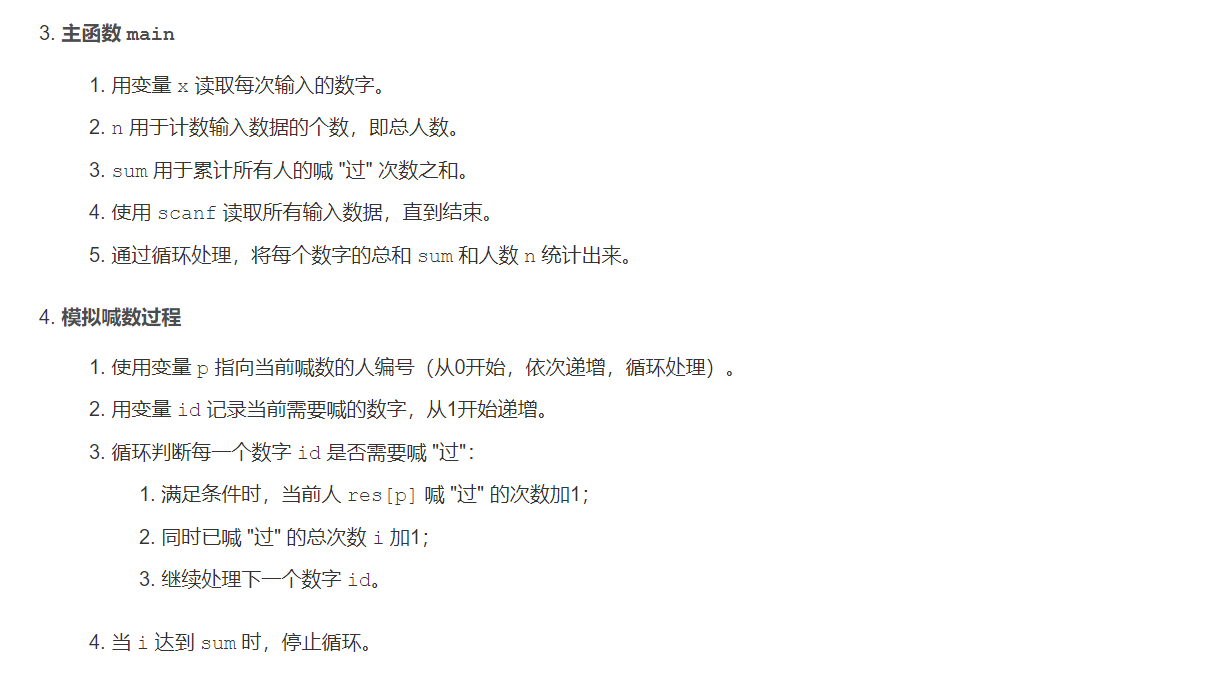


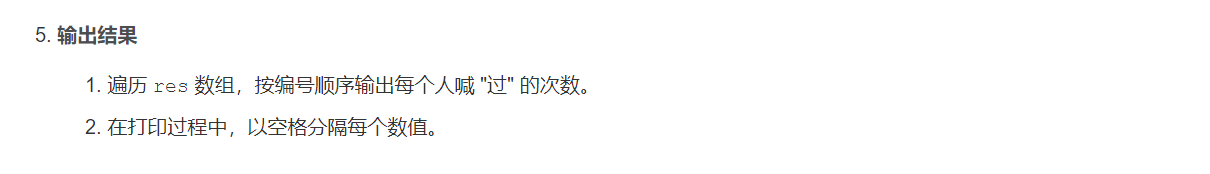
0 0 0 2 1



0 2 0 1 0









import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int[] res = new int[100010]; // 用于存储每个人喊"过"的次数

int n = 0; // 用于统计总人数，即数组长度

int sum = 0; // 用于存储所有喊过的总次数

// 读取输入的每个数字，直到文件结束

while (scanner.hasNextInt()) {

int x = scanner.nextInt();

n++; // 统计总人数

sum += x; // 统计所有喊过的总次数

}

int p = 0; // 指向当前喊数的人的编号（0到n-1）

int id = 1; // 当前需要喊的数字

// 模拟从1开始依次喊数，直到累计喊"过"的次数等于sum

for (int i = 0; i < sum; ) {

if (judge(id)) { // 判断当前数字是否需要喊"过"

res[p]++; // 当前人喊"过"的次数加1

i++; // 已喊"过"的次数加1

}

id++; // 准备喊下一个数字

p = (p + 1) % n; // 轮到下一个人喊数（顺时针轮转）

}

// 输出每个人的喊"过"次数，按照正确的顺序

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

if (i != 0) System.out.print(" ");

System.out.print(res[i]);

}

}

/\*\*

\* 判断数字x是否需要喊"过"

\* 满足以下条件之一：

\* 1. 数字是7的倍数

\* 2. 数字中含有7

\* @param x 需要判断的数字

\* @return 若需要喊"过"，返回true，否则返回false

\*/

public static boolean judge(int x) {

if (x % 7 == 0) return true; // 是7的倍数

while (x > 0) {

if (x % 10 == 7) return true; // 数字中含有7

x /= 10;

}

return false;

}

}



def judge(x):

"""

判断数字x是否需要喊"过"

满足以下条件之一：

1. 数字是7的倍数

2. 数字中含有7

:param x: 需要判断的数字

:return: 若需要喊"过"，返回True，否则返回False

"""

if x % 7 == 0:

return True # 是7的倍数

while x > 0:

if x % 10 == 7:

return True # 数字中含有7

x //= 10

return False

def main():

import sys

input = sys.stdin.read

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

n = len(data) # 总人数

sum\_ = sum(data) # 所有喊过的总次数

res = [0] \* 100010 # 用于存储每个人喊"过"的次数

p = 0 # 指向当前喊数的人的编号（0到n-1）

id = 1 # 当前需要喊的数字

# 模拟从1开始依次喊数，直到累计喊"过"的次数等于sum

i=0

while i<sum\_:

if judge(id):

res[p] += 1 # 当前人喊"过"的次数加1

i += 1 # 准备喊下一个数字

id += 1

p = (p + 1) % n # 轮到下一个人喊数（顺时针轮转）

# 输出每个人的喊"过"次数，按照正确的顺序

print(" ".join(map(str, res[:n])))

if \_\_name\_\_ == "\_\_main\_\_":

main()



#include <iostream>

#include <vector>

using namespace std;

#define MAXN 100010

int res[MAXN]; // 用于存储每个人喊"过"的次数

/\*\*

\* 判断数字x是否需要喊"过"

\* 满足以下条件之一：

\* 1. 数字是7的倍数

\* 2. 数字中含有7

\* @param x 需要判断的数字

\* @return 若需要喊"过"，返回true，否则返回false

\*/

bool judge(int x) {

if (x % 7 == 0) return true; // 是7的倍数

while (x > 0) {

if (x % 10 == 7) return true; // 数字中含有7

x /= 10;

}

return false;

}

int main() {

int x; // 用于存储每次读取的数字

int n = 0; // 用于统计总人数，即数组长度

int sum = 0; // 用于存储所有喊过的总次数

// 读取输入的每个数字，直到文件结束

while (cin >> x) {

n++; // 统计总人数

sum += x; // 统计所有喊过的总次数

}

int p = 0; // 指向当前喊数的人的编号（0到n-1）

int id = 1; // 当前需要喊的数字

// 模拟从1开始依次喊数，直到累计喊"过"的次数等于sum

for (int i = 0; i < sum; ) {

if (judge(id)) { // 判断当前数字是否需要喊"过"

res[p]++; // 当前人喊"过"的次数加1

i++; // 已喊"过"的次数加1

}

id++; // 准备喊下一个数字

p = (p + 1) % n; // 轮到下一个人喊数（顺时针轮转）

}

// 输出每个人的喊"过"次数，按照正确的顺序

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

if (i != 0) cout << " ";

cout << res[i];

}

return 0; // 程序结束

}



#include <stdio.h>

#include <stdbool.h>

#define MAXN 100010

int res[MAXN]; // 用于存储每个人喊"过"的次数

/\*\*

\* 判断数字x是否需要喊"过"

\* 满足以下条件之一：

\* 1. 数字是7的倍数

\* 2. 数字中含有7

\* @param x 需要判断的数字

\* @return 若需要喊"过"，返回true，否则返回false

\*/

bool judge(int x) {

if (x % 7 == 0) return true; // 是7的倍数

while (x > 0) {

if (x % 10 == 7) return true; // 数字中含有7

x /= 10;

}

return false;

}

int main() {

int x; // 用于存储每次读取的数字

int n = 0; // 用于统计总人数，即数组长度

int sum = 0; // 用于存储所有喊过的总次数

// 读取输入的每个数字，直到文件结束

while (scanf("%d", &x) != EOF) {

n++; // 统计总人数

sum += x; // 统计所有喊过的总次数

}

int p = 0; // 指向当前喊数的人的编号（0到n-1）

int id = 1; // 当前需要喊的数字

// 模拟从1开始依次喊数，直到累计喊"过"的次数等于sum

for (int i = 0; i < sum; ) {

if (judge(id)) { // 判断当前数字是否需要喊"过"

res[p]++; // 当前人喊"过"的次数加1

i++; // 已喊"过"的次数加1

}

id++; // 准备喊下一个数字

p = (p + 1) % n; // 轮到下一个人喊数（顺时针轮转）

}

// 输出每个人的喊"过"次数，按照正确的顺序

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

if (i != 0) printf(" ");

printf("%d", res[i]);

}

return 0; // 程序结束

}



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

let input = [];

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

input.push(line);

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

const data = input[0].split(" ").map(Number);

const n = data.length; // 总人数

const sum = data.reduce((a, b) => a + b, 0); // 所有喊过的总次数

const res = Array(100010).fill(0); // 用于存储每个人喊"过"的次数

let p = 0; // 指向当前喊数的人的编号（0到n-1）

let id = 1; // 当前需要喊的数字

// 模拟从1开始依次喊数，直到累计喊"过"的次数等于sum

for (let i = 0; i < sum; ) {

if (judge(id)) {

// 判断当前数字是否需要喊"过"

res[p]++; // 当前人喊"过"的次数加1

i++; // 已喊"过"的次数加1

}

id++; // 准备喊下一个数字

p = (p + 1) % n; // 轮到下一个人喊数（顺时针轮转）

}

// 输出每个人的喊"过"次数，按照正确的顺序

console.log(res.slice(0, n).join(" "));

});

/\*\*

\* 判断数字x是否需要喊"过"

\* 满足以下条件之一：

\* 1. 数字是7的倍数

\* 2. 数字中含有7

\* @param {number} x 需要判断的数字

\* @return {boolean} 若需要喊"过"，返回true，否则返回false

\*/

function judge(x) {

if (x % 7 === 0) return true; // 是7的倍数

while (x > 0) {

if (x % 10 === 7) return true; // 数字中含有7

x = Math.floor(x / 10);

}

return false;

}



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

func main() {

reader := bufio.NewReader(os.Stdin)

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

data := strings.Fields(input)

n := len(data) // 总人数

sum := 0 // 所有喊过的总次数

res := make([]int, 100010) // 用于存储每个人喊"过"的次数

for \_, str := range data {

x, \_ := strconv.Atoi(str)

sum += x // 统计所有喊过的总次数

}

p := 0 // 指向当前喊数的人的编号（0到n-1）

id := 1 // 当前需要喊的数字

// 模拟从1开始依次喊数，直到累计喊"过"的次数等于sum

for i := 0; i < sum; {

if judge(id) { // 判断当前数字是否需要喊"过"

res[p]++ // 当前人喊"过"的次数加1

i++ // 已喊"过"的次数加1

}

id++ // 准备喊下一个数字

p = (p + 1) % n // 轮到下一个人喊数（顺时针轮转）

}

// 输出每个人的喊"过"次数，按照正确的顺序

for i := 0; i < n; i++ {

if i != 0 {

fmt.Print(" ")

}

fmt.Print(res[i])

}

fmt.Println()

}

/\*\*

\* 判断数字x是否需要喊"过"

\* 满足以下条件之一：

\* 1. 数字是7的倍数

\* 2. 数字中含有7

\* @param x 需要判断的数字

\* @return 若需要喊"过"，返回true，否则返回false

\*/

func judge(x int) bool {

if x%7 == 0 {

return true // 是7的倍数

}

for x > 0 {

if x%10 == 7 {

return true // 数字中含有7

}

x /= 10

}

return false

}