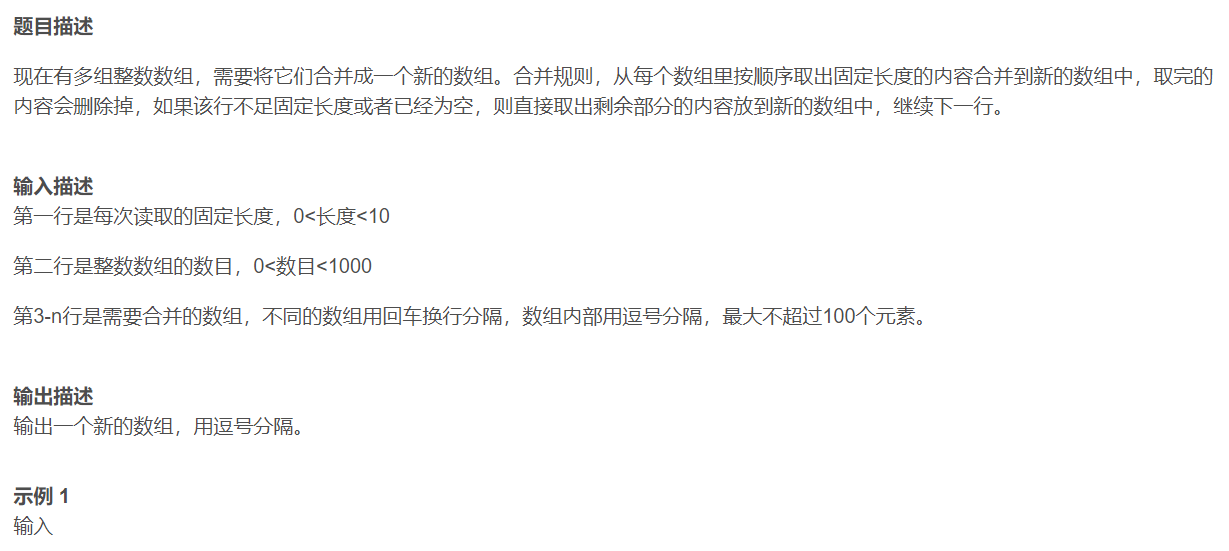
# **E卷-数组拼接[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**



3

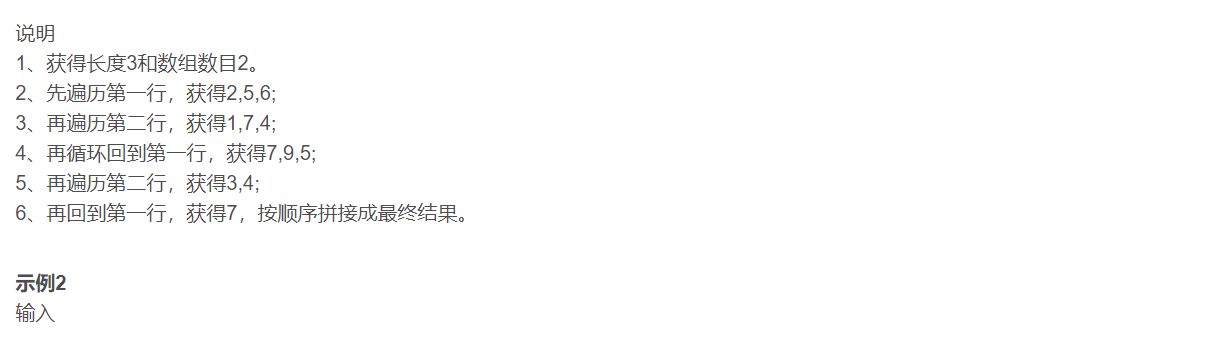
2

2,5,6,7,9,5,7

1,7,4,3,4



2,5,6,1,7,4,7,9,5,3,4,7



4

3

1,2,3,4,5,6

1,2,3

1,2,3,4



1,2,3,4,1,2,3,1,2,3,4,5,6



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 读取每次提取的固定长度 k

int k = scanner.nextInt();

// 读取数组的数目 n

int n = scanner.nextInt();

scanner.nextLine(); // 读取行尾的换行符

// 用于存储所有的数组，每个数组使用 LinkedList（双端队列）存储

List<Deque<Integer>> arrays = new ArrayList<>();

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

Deque<Integer> deque = new LinkedList<>();

// 读取一行并用逗号分隔，将其转换为整数并存入双端队列

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

for (String part : parts) {

deque.add(Integer.parseInt(part));

}

arrays.add(deque);

}

// 用于存储最终合并后的结果数组

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

// 无限循环，直到所有数组都处理完

while (true) {

boolean done = true; // 标识是否所有数组都已经处理完毕

// 遍历每一个数组

for (Deque<Integer> deque : arrays) {

// 从当前数组中取出最多 k 个元素

for (int j = 0; j < k && !deque.isEmpty(); j++) {

// 将取出的元素添加到结果数组中

res.add(deque.poll());

done = false; // 标识仍有数组未处理完

}

}

if (done) {

break; // 如果所有数组都处理完毕，退出循环

}

}

// 输出最终合并后的结果数组，元素用逗号分隔

System.out.println(String.join(",",

res.stream().map(String::valueOf).toArray(String[]::new)));

}

}



from collections import deque

# 读取每次要从每个数组中提取的长度

k = int(input())

# 读取总的数组数目

n = int(input())

# 用于存储所有的数组，每个数组使用双端队列存储

a = []

for \_ in range(n):

# 读取一个数组，并用逗号分隔，将其转换为整数并存入双端队列

now = list(map(int, input().split(",")))

a.append(deque(now))

# 用于存储最终合并后的结果数组

res = []

# 无限循环，直到所有数组都处理完

while True:

done = 1 # 标识是否所有数组都已经处理完毕

# 遍历每一个数组

for i in range(n):

# 从当前数组中取出最多k个元素

for j in range(min(k, len(a[i]))):

# 将取出的元素添加到结果数组中

res.append(a[i].popleft())

done = 0 # 标识仍有数组未处理完

if done:

break # 如果所有数组都处理完毕，退出循环

# 输出最终合并后的结果数组，元素用逗号分隔

print(",".join(map(str, res)))



#include <iostream>

#include <vector>

#include <deque>

#include <string>

#include <sstream>

using namespace std;

int main() {

int k, n;

// 读取每次提取的固定长度 k

cin >> k;

// 读取数组的数目 n

cin >> n;

cin.ignore(); // 读取行尾的换行符

// 用于存储所有的数组，每个数组使用 deque（双端队列）存储

vector<deque<int>> arrays(n);

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

string line;

getline(cin, line);

istringstream iss(line);

string item;

while (getline(iss, item, ',')) {

arrays[i].push\_back(stoi(item));

}

}

// 用于存储最终合并后的结果数组

vector<int> res;

// 无限循环，直到所有数组都处理完

while (true) {

bool done = true; // 标识是否所有数组都已经处理完毕

// 遍历每一个数组

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

// 从当前数组中取出最多 k 个元素

for (int j = 0; j < k && !arrays[i].empty(); ++j) {

// 将取出的元素添加到结果数组中

res.push\_back(arrays[i].front());

arrays[i].pop\_front();

done = false; // 标识仍有数组未处理完

}

}

if (done) {

break; // 如果所有数组都处理完毕，退出循环

}

}

// 输出最终合并后的结果数组，元素用逗号分隔

for (size\_t i = 0; i < res.size(); ++i) {

if (i > 0) cout << ",";

cout << res[i];

}

cout << endl;

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_SIZE 1000

// 函数用于分割字符串

int split(char\* str, char\* delimiter, int\* array) {

char\* token;

int count = 0;

token = strtok(str, delimiter);

while (token != NULL) {

array[count++] = atoi(token);

token = strtok(NULL, delimiter);

}

return count;

}

int main() {

int k, n;

// 读取每次提取的固定长度 k

scanf("%d", &k);

// 读取数组的数目 n

scanf("%d", &n);

getchar(); // 读取行尾的换行符

// 用于存储所有的数组

int arrays[MAX\_SIZE][MAX\_SIZE] = {0};

int lengths[MAX\_SIZE] = {0};

int pos[MAX\_SIZE] = {0};

// 读取每个数组并存储

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

char line[5000]; // 假设一行最多5000字符

fgets(line, 5000, stdin);

line[strcspn(line, "\n")] = '\0'; // 移除换行符

lengths[i] = split(line, ",", arrays[i]);

}

// 用于存储最终合并后的结果数组

int res[MAX\_SIZE] = {0};

int res\_index = 0;

// 无限循环，直到所有数组都处理完

while (1) {

int done = 1; // 标识是否所有数组都已经处理完毕

// 遍历每一个数组

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

// 从当前数组中取出最多 k 个元素

for(int j=0;j<k&&pos[i]!=lengths[i];++j){

res[res\_index++]=arrays[i][pos[i]];

pos[i]+=1;

done=0;

}

}

if (done) {

break; // 如果所有数组都处理完毕，退出循环

}

}

// 输出最终合并后的结果数组，元素用逗号分隔

for (int i = 0; i < res\_index; i++) {

if (i > 0) printf(",");

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

}

printf("\n");

return 0;

}



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

let input = [];

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

input.push(line);

});

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

// 读取每次提取的固定长度 k

const k = parseInt(input[0]);

// 读取数组的数目 n

const n = parseInt(input[1]);

// 用于存储所有的数组，每个数组使用数组存储

const arrays = [];

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

arrays.push(input[i + 2].split(",").map(Number));

}

// 用于存储最终合并后的结果数组

const res = [];

// 无限循环，直到所有数组都处理完

while (true) {

let done = true; // 标识是否所有数组都已经处理完毕

// 遍历每一个数组

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

// 从当前数组中取出最多 k 个元素

let count = 0;

while (count < k && arrays[i].length > 0) {

// 将取出的元素添加到结果数组中

res.push(arrays[i].shift());

count++;

done = false; // 标识仍有数组未处理完

}

}

if (done) {

break; // 如果所有数组都处理完毕，退出循环

}

}

// 输出最终合并后的结果数组，元素用逗号分隔

console.log(res.join(","));

});



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

func main() {

reader := bufio.NewReader(os.Stdin)

// 读取每次提取的固定长度 k

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

kStr = strings.TrimSpace(kStr)

k, \_ := strconv.Atoi(kStr)

// 读取数组的数目 n

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

nStr = strings.TrimSpace(nStr)

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

// 用于存储所有的数组，每个数组使用 slice 存储

arrays := make([][]int, n)

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

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

line = strings.TrimSpace(line)

strNums := strings.Split(line, ",")

for \_, strNum := range strNums {

num, \_ := strconv.Atoi(strNum)

arrays[i] = append(arrays[i], num)

}

}

// 用于存储最终合并后的结果数组

var res []int

// 无限循环，直到所有数组都处理完

for {

done := true // 标识是否所有数组都已经处理完毕

// 遍历每一个数组

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

// 从当前数组中取出最多 k 个元素

count := 0

for count < k && len(arrays[i]) > 0 {

// 将取出的元素添加到结果数组中

res = append(res, arrays[i][0])

arrays[i] = arrays[i][1:]

count++

done = false // 标识仍有数组未处理完

}

}

if done {

break // 如果所有数组都处理完毕，退出循环

}

}

// 输出最终合并后的结果数组，元素用逗号分隔

fmt.Println(strings.Join(intSliceToStringSlice(res), ","))

}

// 辅助函数：将整数切片转换为字符串切片

func intSliceToStringSlice(ints []int) []string {

strs := make([]string, len(ints))

for i, v := range ints {

strs[i] = strconv.Itoa(v)

}

return strs

}