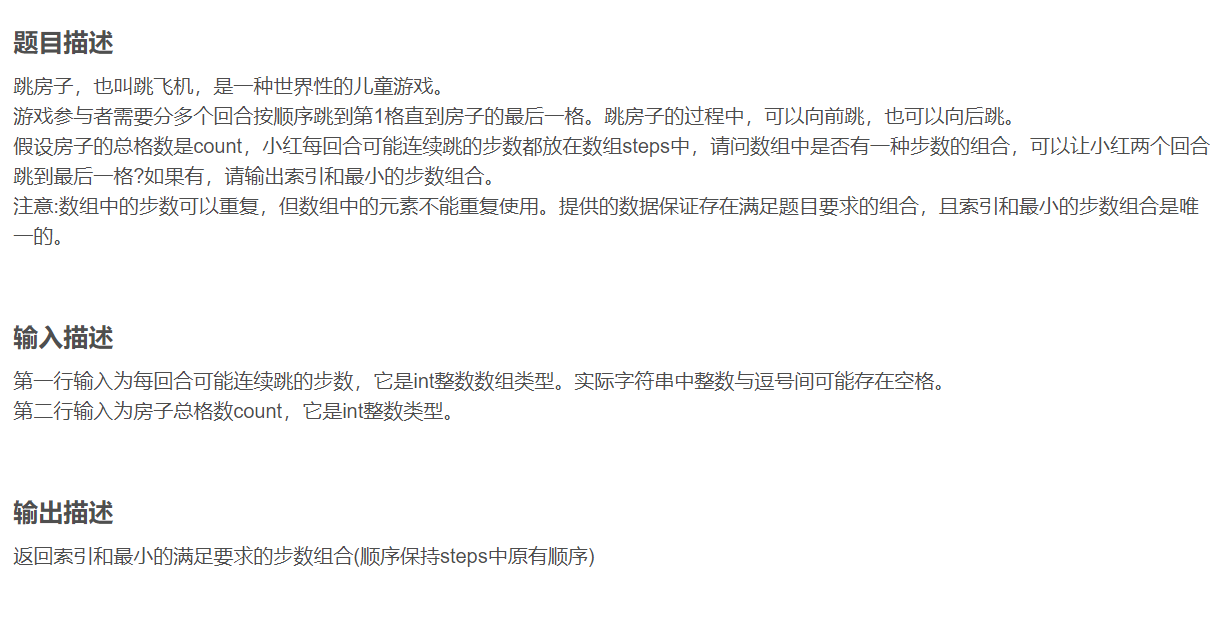
# **E卷-跳房子1[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**









import java.util.Scanner;

public class Main {

public static void main(String[] args) {

// 创建一个扫描器对象，用于读取输入

Scanner scanner = new Scanner(System.in);

// 读取第一行输入，表示每回合可能连续跳的步数

String input = scanner.nextLine();

// 读取第二行输入，表示房子总格数

long k = scanner.nextLong();

// 去掉输入字符串中的方括号和空格

input = input.replace("[", "").replace("]", "").replace(" ", "");

// 按逗号分割字符串，得到步数数组的字符串表示

String[] ss = input.split(",");

// 创建一个长整型数组，用于存储步数

long[] a = new long[ss.length];

// 将字符串数组转换为长整型数组

for (int i = 0; i < ss.length; i++) {

a[i] = Long.parseLong(ss[i].trim());

}

// 初始化一个较大的值，用于记录最小的索引和

int mx = 999999;

// 初始化两个变量，用于记录满足条件的步数组合

long x = 0, y = 0;

// 获取步数数组的长度

int n = a.length;

// 双重循环遍历步数数组，寻找满足条件的步数组合

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

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

// 如果两个步数之和等于房子总格数

if (a[i] + a[j] == k) {

// 计算当前组合的索引和

int s = i + j;

// 如果当前组合的索引和小于记录的最小索引和

if (i + j < mx) {

// 更新最小索引和

mx = i + j;

// 更新满足条件的步数组合

x = a[i];

y = a[j];

}

}

}

}

// 输出满足条件的步数组合

System.out.print("[" + x + "," + y + "]");

}

}



# 导入sys模块，以便从标准输入读取输入

import sys

# 从标准输入读取输入

input = sys.stdin.read()

# 按换行符分割输入，得到每行数据

lines = input.split("\n")

# 移除行首和行尾的空白符，解析步数数组字符串

steps\_str = lines[0].replace("[", "").replace("]", "").replace(" ", "")

# 按逗号分割步数字符串，转换为整数数组

steps = list(map(int, steps\_str.split(",")))

# 读取房子总格数，并转换为整数

count = int(lines[1])

# 初始化一个很大的值，用于记录最小的索引和

min\_index\_sum = float("inf")

# 初始化一个空列表，用于保存最终的步数组合

result = []

# 双重循环遍历步数数组，找出满足条件的步数组合

for i in range(len(steps)):

for j in range(i + 1, len(steps)):

# 如果两个步数之和等于房子总格数

if steps[i] + steps[j] == count:

# 计算当前组合的索引和

index\_sum = i + j

# 如果当前组合的索引和小于记录的最小索引和

if index\_sum < min\_index\_sum:

# 更新最小索引和

min\_index\_sum = index\_sum

# 更新满足条件的步数组合

result = [steps[i], steps[j]]

# 输出满足条件的步数组合

print(str(result).replace(" ",""))



#include<bits/stdc++.h>

using namespace std;

int main() {

// 读取步数数组和房子总格数的输入

string input\_steps, count\_str;

getline(cin, input\_steps);

getline(cin, count\_str);

// 去掉方括号和空格，将字符串按逗号分割为步数

input\_steps.erase(remove(input\_steps.begin(), input\_steps.end(), '['),

input\_steps.end());

input\_steps.erase(remove(input\_steps.begin(), input\_steps.end(), ']'),

input\_steps.end());

input\_steps.erase(remove(input\_steps.begin(), input\_steps.end(), ' '),

input\_steps.end());

// 创建一个字符串流来处理步数字符串

stringstream ss(input\_steps);

vector<int> steps;

string temp;

// 将字符串中的步数转换为整数并加入到数组中

while (getline(ss, temp, ',')) {

steps.push\_back(stoi(temp));

}

// 将房子总格数转换为整数

int count = stoi(count\_str);

// 初始化最小索引和为最大整数

int min\_index\_sum = INT\_MAX;

// 初始化两个变量来记录满足条件的步数组合

int result\_x = 0, result\_y = 0;

// 双重循环遍历步数数组，寻找满足条件的步数组合

for (int i = 0; i < steps.size(); ++i) {

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

// 如果两个步数之和等于房子总格数

if (steps[i] + steps[j] == count) {

// 计算当前组合的索引和

int index\_sum = i + j;

// 如果当前组合的索引和小于记录的最小索引和

if (index\_sum < min\_index\_sum) {

// 更新最小索引和

min\_index\_sum = index\_sum;

// 更新满足条件的步数组合

result\_x = steps[i];

result\_y = steps[j];

}

}

}

}

// 输出满足条件的步数组合

cout << "[" << result\_x << "," << result\_y << "]" << endl;

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <limits.h>

int main() {

// 读取步数数组和房子总格数的输入

char input\_steps[10000];

char count\_str[10];

fgets(input\_steps, sizeof(input\_steps), stdin);

fgets(count\_str, sizeof(count\_str), stdin);

// 去掉方括号和空格，将字符串按逗号分割为步数

char\* p1 = input\_steps;

while (\*p1) {

if (\*p1 == '[' || \*p1 == ']' || \*p1 == ' ') {

memmove(p1, p1 + 1, strlen(p1));

} else {

p1++;

}

}

// 创建步数数组

int steps[1000];

int steps\_count = 0;

char\* token = strtok(input\_steps, ",");

while (token != NULL) {

steps[steps\_count++] = atoi(token);

token = strtok(NULL, ",");

}

// 将房子总格数转换为整数

int count = atoi(count\_str);

// 初始化最小索引和为最大整数

int min\_index\_sum = INT\_MAX;

// 初始化两个变量来记录满足条件的步数组合

int result\_x = 0, result\_y = 0;

// 双重循环遍历步数数组，寻找满足条件的步数组合

for (int i = 0; i < steps\_count; ++i) {

for (int j = i + 1; j < steps\_count; ++j) {

// 如果两个步数之和等于房子总格数

if (steps[i] + steps[j] == count) {

// 计算当前组合的索引和

int index\_sum = i + j;

// 如果当前组合的索引和小于记录的最小索引和

if (index\_sum < min\_index\_sum) {

// 更新最小索引和

min\_index\_sum = index\_sum;

// 更新满足条件的步数组合

result\_x = steps[i];

result\_y = steps[j];

}

}

}

}

// 输出满足条件的步数组合

printf("[%d,%d]\n", result\_x, result\_y);

return 0;

}



// 在标准输入和输出之间创建读写流

const readline = require("readline");

// 创建接口以便读取输入

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

let input = [];

// 读取输入的行

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

input.push(line.trim());

});

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

// 将输入转为整数并赋值给 x 和 y

let steps = input[0]

.replace("[", "")

.replace("]", "")

.replace(/\s+/g, "")

.split(",")

.map(Number);

let count = parseInt(input[1]);

// 初始化一个较大的值，用于记录最小的索引和

let minIndexSum = Number.MAX\_SAFE\_INTEGER;

let result = [];

// 双重循环遍历步数数组，寻找满足条件的步数组合

for (let i = 0; i < steps.length; i++) {

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

// 如果两个步数之和等于房子总格数

if (steps[i] + steps[j] === count) {

// 计算当前组合的索引和

let indexSum = i + j;

// 如果当前组合的索引和小于记录的最小索引和

if (indexSum < minIndexSum) {

// 更新最小索引和

minIndexSum = indexSum;

// 更新满足条件的步数组合

result = [steps[i], steps[j]];

}

}

}

}

// 输出满足条件的步数组合

console.log(`[${result[0]},${result[1]}]`);

});



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

func main() {

// 创建一个读取器，从标准输入读取数据

reader := bufio.NewReader(os.Stdin)

// 读取第一行输入

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

line = strings.TrimSpace(line)

line = strings.Replace(line, "[", "", -1)

line = strings.Replace(line, "]", "", -1)

line = strings.Replace(line, " ", "", -1)

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

// 将读取的字符串转为整数数组

steps := make([]int, len(parts))

for i, part := range parts {

steps[i], \_ = strconv.Atoi(part)

}

// 读取第二行输入

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

count, \_ := strconv.Atoi(strings.TrimSpace(line))

// 初始化一个较大的值，用于记录最小的索引和

minIndexSum := int(^uint(0) >> 1) // 最大的 int 值

var result [2]int

// 双重循环遍历步数数组，寻找满足条件的步数组合

for i := 0; i < len(steps); i++ {

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

// 如果两个步数之和等于房子总格数

if steps[i]+steps[j] == count {

// 计算当前组合的索引和

indexSum := i + j

// 如果当前组合的索引和小于记录的最小索引和

if indexSum < minIndexSum {

// 更新最小索引和

minIndexSum = indexSum

// 更新满足条件的步数组合

result[0] = steps[i]

result[1] = steps[j]

}

}

}

}

// 输出满足条件的步数组合

fmt.Printf("[%d,%d]\n", result[0], result[1])

}