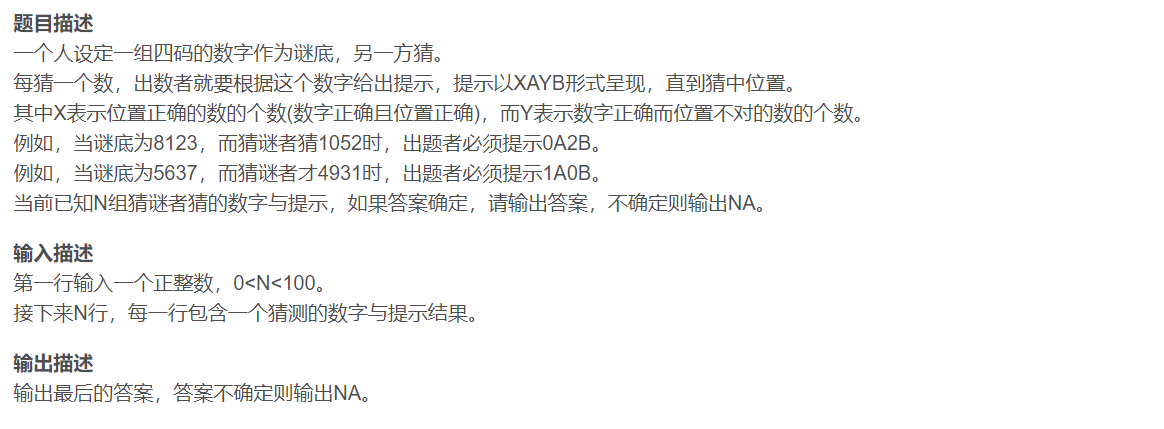
# **E卷-猜数字[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**







import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// 读取输入的组数

int n = sc.nextInt();

sc.nextLine(); // 读取余下的行

List<String[]> guesses = new ArrayList<>();

// 读取每组猜测和提示

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

String line = sc.nextLine();

String[] parts = line.split(" ");

guesses.add(new String[] {parts[0], parts[1]});

}

// 初始化所有可能的四位数

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

for (int i = 0; i < 10000; i++) {

res.add(i);

}

// 检查每个猜测和提示

for (String[] guess : guesses) {

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

for (int j : res) {

// 如果当前数字符合提示，则保留

if (get(j, guess[0]).equals(guess[1])) {

now.add(j);

}

}

// 更新可能的结果

res = now;

}

// 输出结果

if (res.size() == 1) {

System.out.println(res.get(0));

} else {

System.out.println("NA");

}

}

private static String get(int x, String y) {

// 确保x是四位数

String sx = String.format("%04d", x);

int both\_ok = 0; // 位置和数字都正确的个数

int num\_ok = 0; // 数字正确但位置不对的个数

int[] a = new int[10]; // 记录x中各数字的个数

int[] b = new int[10]; // 记录y中各数字的个数

// 检查每一位数是否位置和数字都正确，并统计数字出现次数

for (int i = 0; i < 4; i++) {

if (sx.charAt(i) == y.charAt(i)) {

both\_ok++;

} else {

a[sx.charAt(i) - '0']++;

b[y.charAt(i) - '0']++;

}

}

// 计算数字正确但位置不对的个数

for (int i = 0; i < 10; i++) {

num\_ok += Math.min(a[i], b[i]);

}

return both\_ok + "A" + num\_ok + "B";

}

}



def get(x, y):

x = str(x)

# 确保x是4位数字，不足的前面补0

while len(x) < 4:

x = "0" + x

both\_ok = 0 # X: 位置和数字都正确的个数

num\_ok = 0 # Y: 数字正确但位置不对的个数

a = [0] \* 10 # 记录x中各数字的个数

b = [0] \* 10 # 记录y中各数字的个数

# 检查每一位数是否位置和数字都正确，并统计数字出现次数

for i in range(4):

if x[i] == y[i]:

both\_ok += 1

else:

a[int(x[i])] += 1

b[int(y[i])] += 1

# 计算数字正确但位置不对的个数

for i in range(10):

num\_ok += min(a[i], b[i])

return str(both\_ok) + "A" + str(num\_ok) + "B"

# 读取输入

n = int(input())

a = []

for \_ in range(n):

x, y = input().split()

a.append((x, y))

# 初始化所有可能的四位数

res = [i for i in range(10000)]

for i in a:

now = []

# 筛选符合所有提示条件的数

for j in res:

if get(j, i[0]) == i[1]:

now.append(j)

res = now[:]

# 如果只有一个结果则输出，否则输出NA

if len(res) == 1:

print(res[0])

else:

print("NA")



#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

using namespace std;

string get(int x, string y) {

string sx = to\_string(x);

// 确保sx是4位数，不足前面补0

while (sx.length() < 4) {

sx = "0" + sx;

}

int both\_ok = 0; // 位置和数字都正确的个数

int num\_ok = 0; // 数字正确但位置不对的个数

int a[10] = {0}; // 记录x中各数字的个数

int b[10] = {0}; // 记录y中各数字的个数

for (int i = 0; i < 4; i++) {

if (sx[i] == y[i]) {

both\_ok++;

} else {

a[sx[i] - '0']++;

b[y[i] - '0']++;

}

}

// 计算数字正确但位置不对的个数

for (int i = 0; i < 10; i++) {

num\_ok += min(a[i], b[i]);

}

return to\_string(both\_ok) + "A" + to\_string(num\_ok) + "B";

}

int main() {

int n;

cin >> n;

cin.ignore(); // 忽略换行符

vector<pair<string, string>> a;

// 读取每组猜测和提示

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

string x, y;

cin >> x >> y;

a.emplace\_back(x, y);

}

// 初始化所有可能的四位数

vector<int> res;

for (int i = 0; i < 10000; i++) {

res.push\_back(i);

}

for (const auto& p : a) {

vector<int> now;

for (int j : res) {

// 如果当前数字符合提示，则保留

if (get(j, p.first) == p.second) {

now.push\_back(j);

}

}

// 更新可能的结果

res = now;

}

// 输出结果

if (res.size() == 1) {

cout << res[0] << endl;

} else {

cout << "NA" << endl;

}

return 0;

}



#include <stdio.h>

#include <string.h>

// 将x转换为4位数的字符串，不足前面补0

void toFourDigits(int x, char\* dst) {

sprintf(dst, "%04d", x);

}

// 计算x和y之间的提示

void get(int x, const char\* y, char\* result) {

char sx[5];

toFourDigits(x, sx);

int both\_ok = 0; // 位置和数字都正确的个数

int num\_ok = 0; // 数字正确但位置不对的个数

int a[10] = {0}; // 记录x中各数字的个数

int b[10] = {0}; // 记录y中各数字的个数

for (int i = 0; i < 4; i++) {

if (sx[i] == y[i]) {

both\_ok++;

} else {

a[sx[i] - '0']++;

b[y[i] - '0']++;

}

}

// 计算数字正确但位置不对的个数

for (int i = 0; i < 10; i++) {

num\_ok += a[i] < b[i] ? a[i] : b[i];

}

sprintf(result, "%dA%dB", both\_ok, num\_ok);

}

int main() {

int n;

scanf("%d", &n);

typedef struct {

char guess[5];

char hint[5];

} Guess;

Guess a[n];

// 读取每组猜测和提示

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

scanf("%4s %4s", a[i].guess, a[i].hint);

}

int res[10000];

int res\_len = 0;

// 初始化所有可能的四位数

for (int i = 0; i < 10000; i++) {

res[res\_len++] = i;

}

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

int now[10000];

int now\_len = 0;

for (int j = 0; j < res\_len; j++) {

char result[10];

// 如果当前数字符合提示，则保留

get(res[j], a[i].guess, result);

if (strcmp(result, a[i].hint) == 0) {

now[now\_len++] = res[j];

}

}

res\_len = now\_len;

memcpy(res, now, now\_len \* sizeof(int));

}

// 输出结果

if (res\_len == 1) {

printf("%04d\n", res[0]);

} else {

printf("NA\n");

}

return 0;

}



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

const questions = [];

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

questions.push(input);

if (questions.length > parseInt(questions[0])) {

rl.close();

}

});

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

const n = parseInt(questions[0]);

const a = [];

// 读取每组猜测和提示

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

const [x, y] = questions[i].split(" ");

a.push([x, y]);

}

// 初始化所有可能的四位数

const res = [];

for (let i = 0; i < 10000; i++) {

res.push(i);

}

// 计算x和y之间的提示

const get = (x, y) => {

let sx = x.toString().padStart(4, "0");

let both\_ok = 0; // 位置和数字都正确的个数

let num\_ok = 0; // 数字正确但位置不对的个数

const a = new Array(10).fill(0); // 记录x中各数字的个数

const b = new Array(10).fill(0); // 记录y中各数字的个数

for (let i = 0; i < 4; i++) {

if (sx[i] === y[i]) {

both\_ok++;

} else {

a[parseInt(sx[i])] += 1;

b[parseInt(y[i])] += 1;

}

}

for (let i = 0; i < 10; i++) {

num\_ok += Math.min(a[i], b[i]);

}

return `${both\_ok}A${num\_ok}B`;

};

for (const [guess, hint] of a) {

const now = [];

for (const j of res) {

// 如果当前数字符合提示，则保留

if (get(j, guess) === hint) {

now.push(j);

}

}

// 更新可能的结果

res.length = 0;

res.push(...now);

}

// 输出结果

if (res.length === 1) {

console.log(res[0]);

} else {

console.log("NA");

}

});



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

// 计算x和y之间的提示

func get(x int, y string) string {

sx := fmt.Sprintf("%04d", x) // 确保x是4位数字

bothOk, numOk := 0, 0

a, b := make([]int, 10), make([]int, 10)

for i := 0; i < 4; i++ {

if sx[i] == y[i] {

bothOk++

} else {

a[sx[i]-'0']++

b[y[i]-'0']++

}

}

for i := 0; i < 10; i++ {

if a[i] < b[i] {

numOk += a[i]

} else {

numOk += b[i]

}

}

return fmt.Sprintf("%dA%dB", bothOk, numOk)

}

func main() {

reader := bufio.NewReader(os.Stdin)

nStr, \_ := reader.ReadString('\n') // 读取输入的组数

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

guesses := make([][]string, n)

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

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

parts := strings.Fields(line)

guesses[i] = parts

}

// 初始化所有可能的四位数

res := make([]int, 10000)

for i := 0; i < 10000; i++ {

res[i] = i

}

for \_, guess := range guesses {

var now []int

for \_, r := range res {

// 如果当前数字符合提示，则保留

if get(r, guess[0]) == guess[1] {

now = append(now, r)

}

}

res = now

}

// 输出结果

if len(res) == 1 {

fmt.Println(res[0])

} else {

fmt.Println("NA")

}

}