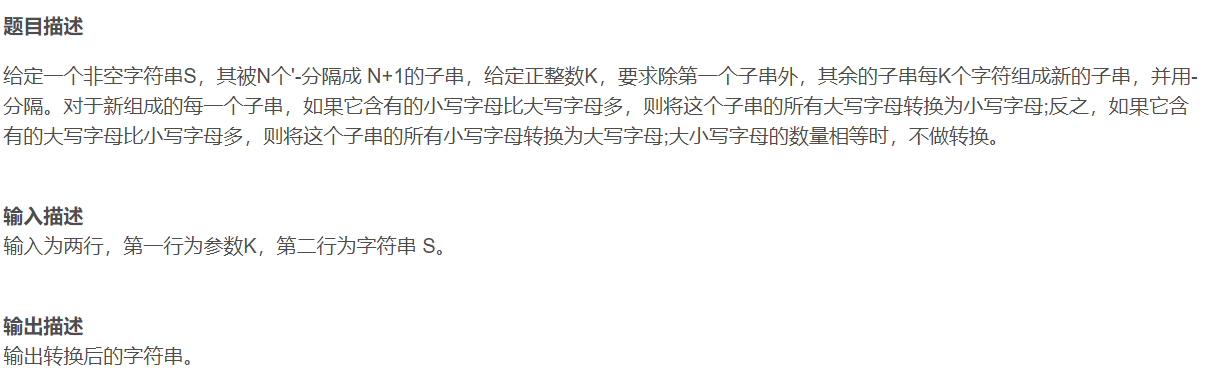
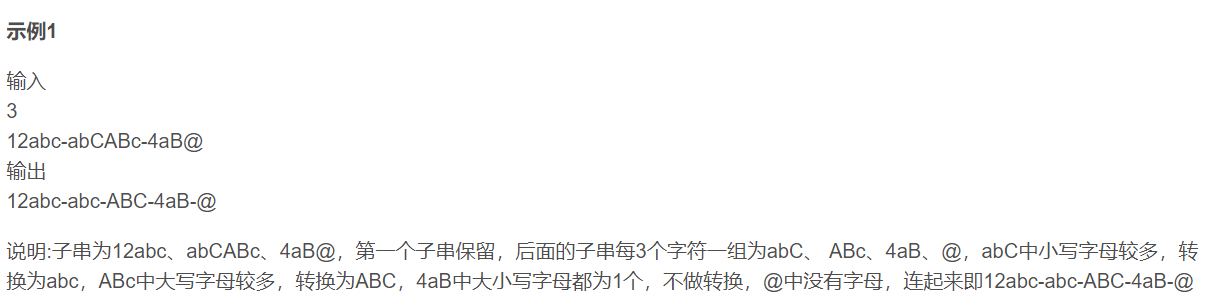
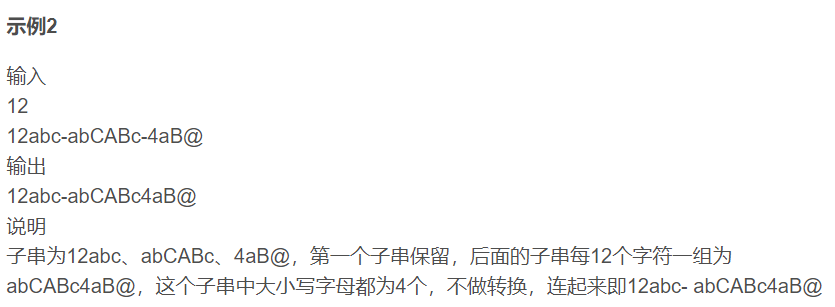
# **E卷-字符串分割[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**









import java.util.Scanner;

public class Main {

// 定义 change 函数

public static String change(String now) {

int cntd = 0, cntx = 0;

// 遍历字符串，计算大写字母和小写字母的数量

for (char x : now.toCharArray()) {

if (Character.isUpperCase(x)) cntd++;

if (Character.isLowerCase(x)) cntx++;

}

// 如果大写字母的数量大于小写字母的数量，返回全大写字符串

if (cntd > cntx) return now.toUpperCase();

// 如果小写字母的数量大于大写字母的数量，返回全小写字符串

if (cntd < cntx) return now.toLowerCase();

// 如果大写字母和小写字母的数量相同，返回原字符串

return now;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 读取输入整数 n

int n = Integer.parseInt(scanner.nextLine());

// 读取一行输入并按 "-" 进行拆分

String s = scanner.nextLine();

String[] parts = s.split("-");

// 拼接分割后的所有部分，形成字符串 ss

StringBuilder ss = new StringBuilder();

for (int i = 1; i < parts.length; i++) {

ss.append(parts[i]);

}

// 初始化计数器 cnt 和临时字符串 t

StringBuilder ans = new StringBuilder(parts[0]);

int cnt = 0;

StringBuilder t = new StringBuilder();

// 输出不需要修改的开头部分

System.out.print(ans);

// 遍历拼接后的字符串 ss

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

cnt++;

t.append(ss.charAt(i));

// 当计数器达到 n 的时候

if (cnt == n) {

System.out.print("-");

System.out.print(change(t.toString()));

cnt = 0;

t.setLength(0); // 清空临时字符串

}

}

// 如果临时字符串 t 不为空，输出处理后的最后一部分内容

if (t.length() > 0) {

System.out.print("-");

System.out.print(change(t.toString()));

}

scanner.close();

}

}



# 读取输入整数 n

n = int(input())

# 读取一行输入并按 "-" 进行拆分，生成列表 s

s = input().split("-")

# 将列表 s 的第二个元素到末尾的所有元素拼接成一个字符串 ss

ss = "".join(s[1:])

# 结果字符串的初始值为列表 s 的第一个元素

ans = s[0]

# 初始化计数器 cnt 和临时字符串 t

cnt, t = 0, ""

# 定义 change 函数，当字母大写多于小写时将字符串转换为大写，当字母小写多于大写时将字符串转换为小写，否则保持不变

def change(now):

cntd, cntx = 0, 0

# 遍历字符串，判断每个字符是大写字母还是小写字母，并进行计数

for x in now:

if ord('Z') >= ord(x) >= ord('A'):

cntd += 1

if ord('z') >= ord(x) >= ord('a'):

cntx += 1

# 如果大写字母的数量大于小写字母的数量，返回全大写字符串

if cntd > cntx:

return now.upper()

# 如果小写字母的数量大于大写字母的数量，返回全小写字符串

if cntd < cntx:

return now.lower()

# 如果大写字母和小写字母的数量相同，返回原字符串

return now

# 打印不需要修改的首个字符串部分

print(s[0], end="")

# 遍历拼接后的字符串 ss

for i in range(len(ss)):

# 计数器加1

cnt += 1

# 将字符添加到临时字符串 t 中

t += ss[i]

# 当计数器达到 n 的时候

if cnt == n:

# 打印一个 "-"

print("-", end="")

# 打印经过 change 函数处理后的临时字符串 t

print(change(t), end="")

# 重置计数器和临时字符串

cnt = 0

t = ""

# 如果临时字符串 t 不为空，说明此时的 t 长度未满 n

if t != "":

# 打印一个 "-"

print("-", end="")

# 打印经过 change 函数处理后的临时字符串 t

print(change(t))



#include <iostream>

#include <string>

#include <cctype> // for isupper and islower

using namespace std;

// 定义 change 函数

string change(string now) {

int cntd = 0, cntx = 0;

// 遍历字符串，计算大写字母和小写字母的数量

for (char x : now) {

if (isupper(x)) cntd++;

if (islower(x)) cntx++;

}

// 如果大写字母的数量大于小写字母的数量，返回全大写字符串

if (cntd > cntx) return now.substr(0, now.length()).assign(now.length(),

toupper(now[0]));

// 如果小写字母的数量大于大写字母的数量，返回全小写字符串

if (cntd < cntx) return now.substr(0, now.length()).assign(now.length(),

tolower(now[0]));

// 如果大写字母和小写字母的数量相同，返回原字符串

return now;

}

int main() {

// 读取输入整数 n

int n;

cin >> n;

cin.ignore(); // 清除缓冲区

// 读取一行输入并按 "-" 进行拆分

string s;

getline(cin, s);

size\_t pos = s.find('-');

string ans = s.substr(0, pos);

string ss = s.substr(pos + 1);

// 初始化计数器 cnt 和临时字符串 t

int cnt = 0;

string t = "";

// 输出不需要修改的开头部分

cout << ans;

// 遍历拼接后的字符串 ss

for (char ch : ss) {

cnt++;

t += ch;

// 当计数器达到 n 的时候

if (cnt == n) {

cout << "-" << change(t);

cnt = 0;

t = "";

}

}

// 如果临时字符串 t 不为空，输出处理后的最后一部分内容

if (!t.empty()) {

cout << "-" << change(t);

}

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h> // for isupper and islower

// 定义 change 函数

void change(char\* now) {

int cntd = 0, cntx = 0;

char\* p = now;

// 遍历字符串，计算大写字母和小写字母的数量

while (\*p) {

if (isupper(\*p)) cntd++;

if (islower(\*p)) cntx++;

p++;

}

// 如果大写字母的数量大于小写字母的数量，转换为全大写

if (cntd > cntx) {

p = now;

while (\*p) {

\*p = toupper(\*p);

p++;

}

}

// 如果小写字母的数量大于大写字母的数量，转换为全小写

else if (cntd < cntx) {

p = now;

while (\*p) {

\*p = tolower(\*p);

p++;

}

}

// 如果大写字母和小写字母的数量相同，保持不变

}

int main() {

// 读取输入整数 n

int n;

scanf("%d\n", &n); // 注意清除缓冲区

// 读取一行输入并按 "-" 进行拆分

char s[1000];

fgets(s, sizeof(s), stdin);

char\* dash\_pos = strchr(s, '-');

\*dash\_pos = '\0'; // 用空字符分割字符串

char\* ans = s;

char\* ss = dash\_pos + 1;

ss[strlen(ss) - 1] = '\0'; // 移除最后的换行符

// 初始化计数器 cnt 和临时字符串 t

int cnt = 0;

char t[1000] = "";

// 输出不需要修改的开头部分

printf("%s", ans);

// 遍历拼接后的字符串 ss

char\* p = ss;

while (\*p) {

if(\*p=='-'){

p++;

continue;

}

cnt++;

strncat(t, p, 1); // 将字符添加到临时字符串 t

// 当计数器达到 n 的时候

if (cnt == n) {

change(t);

printf("-%s", t);

cnt = 0;

t[0] = '\0'; // 清空字符串

}

p++;

}

// 如果临时字符串 t 不为空，输出处理后的最后一部分内容

if (strlen(t) > 0) {

change(t);

printf("-%s", t);

}

return 0;

}

// 3

// 12abc-abCABc-4aB@

// 12abc-abc-ABC-4aB-@



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

// 定义change函数

function change(now) {

let cntd = 0,

cntx = 0;

// 遍历字符串，计算大写字母和小写字母的数量

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

let x = now[i];

if (x >= "A" && x <= "Z") cntd++;

if (x >= "a" && x <= "z") cntx++;

}

// 如果大写字母的数量大于小写字母的数量，返回全大写字符串

if (cntd > cntx) return now.toUpperCase();

// 如果小写字母的数量大于大写字母的数量，返回全小写字符串

if (cntd < cntx) return now.toLowerCase();

// 如果大写字母和小写字母的数量相同，返回原字符串

return now;

}

// 读取输入 n 和字符串，并处理逻辑

rl.question("", (n) => {

n = parseInt(n);

rl.question("", (s) => {

let parts = s.split("-");

let ss = parts.slice(1).join("");

let ans = parts[0];

let cnt = 0;

let t = "";

process.stdout.write(ans);

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

cnt++;

t += ss[i];

// 当计数器达到 n 的时候

if (cnt === n) {

process.stdout.write("-" + change(t));

cnt = 0;

t = "";

}

}

// 如果临时字符串 t 不为空，输出处理后的最后一部分内容

if (t.length > 0) {

process.stdout.write("-" + change(t));

}

rl.close();

});

});



package main

import (

"bufio"

"fmt"

"os"

"strings"

"unicode"

)

// 定义 change 函数

func change(now string) string {

cntd, cntx := 0, 0

// 遍历字符串，计算大写字母和小写字母的数量

for \_, x := range now {

if unicode.IsUpper(x) {

cntd++

}

if unicode.IsLower(x) {

cntx++

}

}

// 如果大写字母的数量大于小写字母的数量，返回全大写字符串

if cntd > cntx {

return strings.ToUpper(now)

}

// 如果小写字母的数量大于大写字母的数量，返回全小写字符串

if cntd < cntx {

return strings.ToLower(now)

}

// 如果大写字母和小写字母的数量相同，返回原字符串

return now

}

func main() {

scanner := bufio.NewScanner(os.Stdin)

// 读取输入整数 n

scanner.Scan()

var n int

fmt.Sscanf(scanner.Text(), "%d", &n)

// 读取一行输入并按 "-" 进行拆分

scanner.Scan()

s := scanner.Text()

parts := strings.Split(s, "-")

// 拼接分割后的所有部分，形成字符串 ss

ss := strings.Join(parts[1:], "")

ans := parts[0]

// 初始化计数器 cnt 和临时字符串 t

cnt := 0

t := ""

// 输出不需要修改的开头部分

fmt.Print(ans)

// 遍历拼接后的字符串 ss

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

cnt++

t += string(ss[i])

// 当计数器达到 n 的时候

if cnt == n {

fmt.Print("-")

fmt.Print(change(t))

cnt = 0

t = ""

}

}

// 如果临时字符串 t 不为空，输出处理后的最后一部分内容

if t != "" {

fmt.Print("-")

fmt.Print(change(t))

}

}