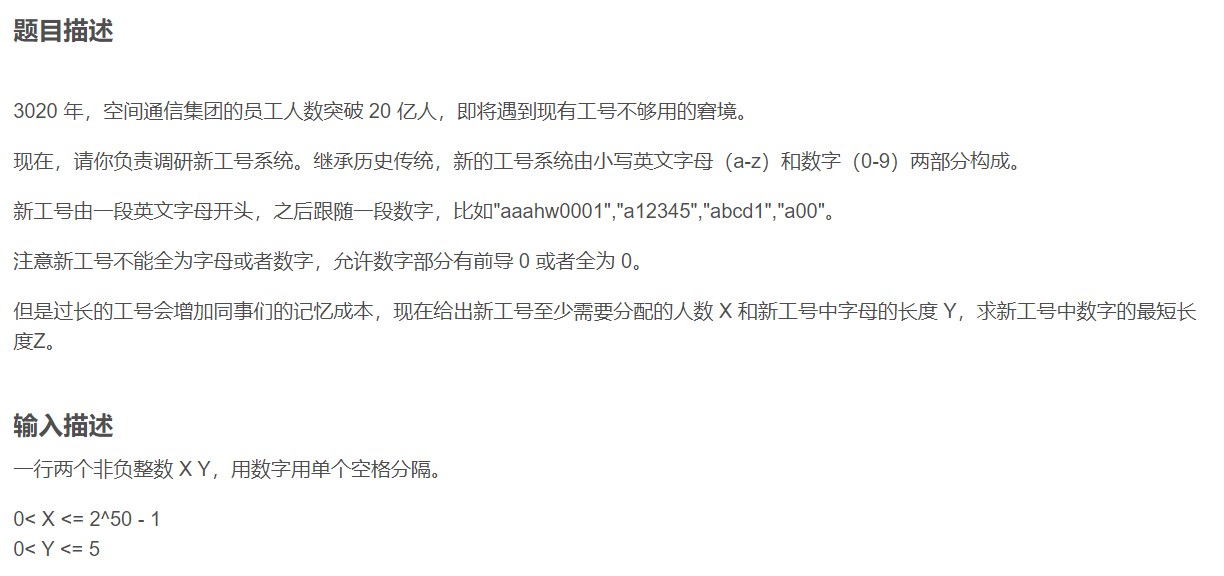
# **E卷-工号不够用了怎么办[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go ）**





260 1



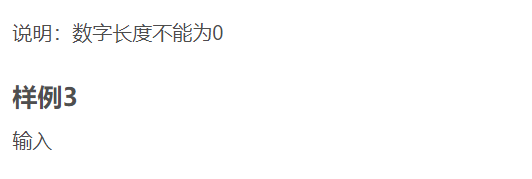
1



26



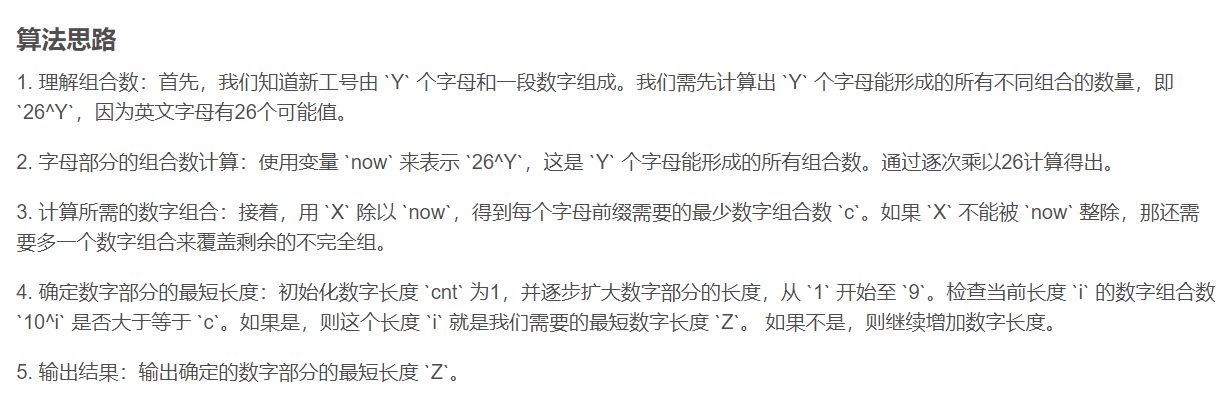
1



2600 1



2





import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// 读取输入的两个整数 X 和 Y

long x = sc.nextLong();

int y = sc.nextInt();

// 计算由 Y 个字母组成的组合数 26^Y

long now = 1;

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

now \*= 26;

}

// 计算每组字母所需的最小数字组合数

long c = x / now;

if (x % now != 0) {

c += 1;

}

// 初始化数字部分的最短长度

long base = 1;

int cnt = 1;

// 确定最短数字长度

for (int i = 1; i < 100; i++) {

base \*= 10;

if (base >= c) {

cnt = i;

break;

}

}

// 输出数字部分的最短长度

System.out.println(cnt);

// 关闭扫描器

sc.close();

}

}



# 读取输入，获取需要分配的人数 X 和字母长度 Y

x, y = map(int, input().split())

# 计算由 Y 个字母组成的工号前缀的组合数

now = 1

for i in range(y):

now \*= 26 # 26个字母的 Y 次方，即组合数为 26^Y

# 计算新工号中数字部分所需的最小组合数

# 每组字母前缀需要多少个数字才能满足所有X个工号

c = x // now # 不整除得到的整数部分

if x % now != 0:

c += 1 # 如果有余数，则需要多一个数字组合

# 初始化数字部分的最短长度，以便逐步计算

base = 1

cnt = 1 # 最短长度从1开始，只包含一位数字

# 计算最小的数字长度Z

for i in range(1, 100):

base \*= 10 # 每增加一位数字，组合数扩大10倍

if base >= c: # 如果当前数字长度的组合数满足c

cnt = i # 确定符合条件的最小的数字长度

break

# 输出最短的数字长度

print(cnt)



#include <iostream>

using namespace std;

int main() {

long long x;

int y;

// 读取输入的两个整数 X 和 Y

cin >> x >> y;

// 计算由 Y 个字母组成的组合数 26^Y

long long now = 1;

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

now \*= 26;

}

// 计算每组字母所需的最小数字组合数

long long c = x / now;

if (x % now != 0) {

c += 1;

}

// 初始化数字部分的最短长度

long long base = 1;

int cnt = 1;

// 确定最短数字长度

for (int i = 1; i < 100; i++) {

base \*= 10;

if (base >= c) {

cnt = i;

break;

}

}

// 输出数字部分的最短长度

cout << cnt << endl;

return 0;

}



#include <stdio.h>

int main() {

long long x;

int y;

// 读取输入的两个整数 X 和 Y

scanf("%lld %d", &x, &y);

// 计算由 Y 个字母组成的组合数 26^Y

long long now = 1;

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

now \*= 26;

}

// 计算每组字母所需的最小数字组合数

long long c = x / now;

if (x % now != 0) {

c += 1;

}

// 初始化数字部分的最短长度

long long base = 1;

int cnt = 1;

// 确定最短数字长度

for (int i = 1; i < 100; i++) {

base \*= 10;

if (base >= c) {

cnt = i;

break;

}

}

// 输出数字部分的最短长度

printf("%d\n", cnt);

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 [x, y] = input[0].split(" ").map(Number);

// 计算由 y 个字母组成的工号前缀的组合数

let now = 1;

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

now \*= 26; // 26 个字母的 y 次方，即组合数为 26^y

}

// 计算新工号中数字部分所需的最小组合数

let c = Math.floor(x / now); // 不整除得到的整数部分

if (x % now !== 0) {

c += 1; // 如果有余数，则需要多一个数字组合

}

// 初始化数字部分的最短长度，以便逐步计算

let base = 1;

let cnt = 1; // 最短长度从 1 开始，只包含一位数字

// 计算最小的数字长度 z

for (let i = 1; i < 100; i++) {

base \*= 10; // 每增加一位数字，组合数扩大 10 倍

if (base >= c) {

cnt = i; // 确定符合条件的最小的数字长度

break;

}

}

// 输出最短的数字长度

console.log(cnt);

});



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)

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

// 将读取的字符串转为整数

x, \_ := strconv.Atoi(parts[0])

y, \_ := strconv.Atoi(parts[1])

// 计算由 Y 个字母组成的工号前缀的组合数

now := 1

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

now \*= 26 // 26 个字母的 Y 次方，即组合数为 26^Y

}

// 计算新工号中数字部分所需的最小组合数

c := x / now // 不整除得到的整数部分

if x%now != 0 {

c += 1 // 如果有余数，则需要多一个数字组合

}

// 初始化数字部分的最短长度，以便逐步计算

base := 1

cnt := 1 // 最短长度从 1 开始，只包含一位数字

// 计算最小的数字长度 Z

for i := 1; i < 100; i++ {

base \*= 10 // 每增加一位数字，组合数扩大 10 倍

if base >= c { // 如果当前数字长度的组合数满足C

cnt = i // 确定符合条件的最小的数字长度

break

}

}

// 输出最短的数字长度

fmt.Println(cnt)

}