# 【华为OD机考 统一考试机试B卷】 全量和已占用字符集 、字符串统计（C++ Java JavaScript Python）

## 华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

2023年11月份，华为官方已经将 华为OD机考：OD统一考试（A卷 / B卷）切换到 OD统一考试（C卷）和 OD统一考试（D卷） 。
真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试( B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境

**题目描述： 全量和已占用字符集 、字符串统计（分值100）**

给定两个字符集合，一个是全量字符集，一个是已占用字符集，已占用字符集中的字符不能再使用。

要求输出剩余可用字符集。

**输入描述**

1. 输入一个字符串 一定包含@，@前为全量字符集 @后的为已占用字符集
2. 已占用字符集中的字符一定是全量字符集中的字符
3. 字符集中的字符跟字符之间使用英文逗号隔开
4. 每个字符都表示为字符+数字的形式用英文冒号分隔，比如a:1标识一个a字符
5. 字符只考虑英文字母，区分大小写
6. 数字只考虑正整型 不超过100
7. 如果一个字符都没被占用 @标识仍存在，例如 a:3,b:5,c:2@

**输出描述**

- 输出可用字符集

- 不同的输出字符集之间用回车换行
- 注意 输出的字符顺序要跟输入的一致，如下面用例不能输出b:3,a:2,c:2
- 如果某个字符已全部占用 则不需要再输出

**ACM输入输出模式**

如果你经常使用**Leetcode**,会知道**letcode**是不需要编写输入输出函数的。但是华为OD机考使用的是 **ACM 模式**，需要手动编写输入和输出。

所以最好在牛-客上提前熟悉这种模式。例如C++使用 `cin/cout` ,python使用 `input()/print()` 。JavaScript使用node的 `readline()` 和 `console.log()` 。Java使用 `sacnner/system.out.print()`

**用例**

输入

```
a:3,b:5,c:2@a:1,b:2
```

输出

```
a:2,b:3,c:2
```

说明

> 全量字符集为三个a，5个b，2个c
> 已占用字符集为1个a，2个b
> 由于已占用字符不能再使用
> 因此剩余可用字符为2个a，3个b，2个c
> 因此输出a:2,b:3,c:2

**机考代码查重**

华为OD机考完成之后，官方会进行代码查重。**华为 od 机考确实有很大的概率抽到原题。**如果碰到了题库中的原题，一定不要直接使用题解中的代码，尤其是变量名，一定要修改，可以改成毫无意义的单词。除了变量名之外，代码的组织结构和逻辑一定要进行改变，这就要求在日常的刷题中，提前编写好属于自己的代码。

**C++**

```cpp
#include <iostream>
#include <vector>
#include <unordered_map>
#include <sstream>

int main() {
    std::string input;
    std::getline(std::cin, input);

    // 将输入字符串按照@符号分割为全量字符集和已占用字符集
    std::string fullCharacterSet = input.substr(0, input.find("@"));
    std::string occupiedCharacterSet = input.substr(input.find("@") + 1);

    // 创建字符列表，用于存储全量字符集中的字符及其对应的数量
    std::vector<std::pair<std::string, int>> characterList;
```

### JavaScript

```javascript
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

rl.on('line', (input) => {
  // 将输入字符串按照@符号分割为全量字符集和已占用字符集
  const splitInput = input.split("@");
  const fullCharacterSet = splitInput[0]; // 全量字符集
  const occupiedCharacterSet = splitInput[1]; // 已占用字符集

  // 创建字符列表，用于存储全量字符集中的字符及其对应的数量
  const characterList = [];
```

### Java

```java
import java.util.Scanner;
import java.util.ArrayList;
import java.util.HashMap;

```

```java
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // 读取输入字符串
        String input = scanner.nextLine();

        // 将输入字符串按照@符号分割为全量字符集和已占用字符集
        String[] splitInput = input.split("@");
        String fullCharacterSet = splitInput[0]; // 全量字符集
        String occupiedCharacterSet = splitInput[1]; // 已占用字符集
```

**Python**

```python
import sys

# 读取输入字符串
input = sys.stdin.readline().strip()

# 将输入字符串按照@符号分割为全量字符集和已占用字符集
splitInput = input.split("@")
fullCharacterSet = splitInput[0] # 全量字符集
occupiedCharacterSet = splitInput[1] # 已占用字符集

# 创建字符列表，用于存储全量字符集中的字符及其对应的数量
characterList = []

# 将全量字符集按照逗号分割为单个字符
fullCharacterSetSplit = fullCharacterSet.split(",")
```

**文章目录**


华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

题目描述： 全量和已占用字符集 、字符串统计（分值100）

输入描述

输出描述

ACM输入输出模式

用例

机考代码查重

C++

JavaScript

Java

Python

完整用例

用例1

用例2

用例3

用例4

用例5

用例6

用例7

用例8

用例9

用例10



**完整用例**

**用例1**

```
a:3,b:5,c:2@a:1,b:2
```

**用例2**

```
a:1,b:2,c:3@d:1,e:2,f:3
```

**用例3**

```
x:10,y:20,z:30@x:5,y:10,z:15
```

**用例4**

```
m:7,n:8,o:9@m:2,n:4,o:6
```

**用例5**

```
x:1,y:2,z:3@x:1,y:1,z:3
```

**用例6**

```
a:10,b:20,c:30@d:1,e:2,f:3
```

**用例7**

```
a:3,b:5,c:2@a:1,b:2
```

**用例8**

```
a:3,b:5,c:2@a:0,b:0,c:0,d:1
```

**用例9**

```
l:1,t:3,u:5,v:7@l:1,t:2,u:3,v:4
```

**用例10**

```
d:5,e:5,f:5@g:2,h:3,i:4
```