题目描述

给出一个仅包含字母的字符串，不包含空格，统计字符串中各个字母（区分大小写）出现的次数，

并按照字母出现次数从大到小的顺序，输出各个字母及其出现次数，

如果次数相同，按照自然顺序进行排序，且小写字母在大写字母之前。

输入描述

输入一行，为一个仅包含字母的字符串。

输出描述

按照字母出现次数从大到小的顺序输出各个字母和字母次数，用英文分号分隔，注意末尾的分号；

字母和次数间用英文冒号分隔。

用例

输入	xyxyXX
输出	x:2;y:2;X:2;
说明	每个字符出现的个数都是2，故x排在y之前，而小写字母x在X之前

输入	abababb
输出	b:4;a:3;
说明	b的出现个数比a多，故b排在a之前

题目解析

本题需要注意的是，题目要求：

> 如果次数相同，按照自然顺序进行排序，且小写字母在大写字母之前。

自然顺序排序，看用例1，应该就是ASCII升序。而后面又要求，小写字母在大写字母之前，小写字母的ASCII码值是大于大小字母的，因此这里又相当于ASCII降序。

因此排序时，如果遇到次数相同的，则需要判断比较的两个字母是否 都为大写 或者 都为小写，若是，则此时按照ASCII码值升序，若不是，则小写字母排在前面，大写字母排在后面。

Java算法源码

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

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String s = sc.nextLine();
    System.out.println(getResult(s));
  }

  public static String getResult(String s) {
    HashMap<Character, Integer> letter = new HashMap<>();

    for (int i = 0; i < s.length(); i++) {
      char c = s.charAt(i);
      letter.put(c, letter.getOrDefault(c, 0) + 1);
    }

    StringBuilder sb = new StringBuilder();

    letter.entrySet().stream()
        .sorted(
            (a, b) -> {
              if (a.getValue() - b.getValue() != 0) {
                return b.getValue() - a.getValue();
              } else {
                if ((isLower(a.getKey()) && isLower(b.getKey()))
                    || (isUpper(a.getKey()) && isUpper(b.getKey()))) {
                  return a.getKey() - b.getKey();
                } else {
                  if (isUpper(a.getKey())) return 1;
                  else return -1;
                }
              }
            })
        .forEach(entry -> sb.append(entry.getKey() + ":" + entry.getValue() + ";"));

    return sb.toString();
  }

  public static boolean isLower(char letter) {
    return letter >= 'a' && letter <= 'z';
  }

  public static boolean isUpper(char letter) {
    return letter >= 'A' && letter <= 'Z';
  }
}
```

JS算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

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

rl.on("line", (line) => {
  console.log(getResult(line));
});

function getResult(s) {
  const letter = {};

  for (let c of s) {
    letter[c] = (letter[c] ?? 0) + 1;
  }

  const arr = [];

  for (let key in letter) {
    arr.push([key, letter[key]]);
  }

  arr.sort((a, b) => {
    if (a[1] != b[1]) return b[1] - a[1];

    if ((isLower(a[0]) && isLower(b[0])) || (isUpper(a[1]) && isUpper(b[1]))) {
      return a[0] > b[0] ? 1 : -1;
    } else {
      if (isUpper(a[0])) return 1;
      else return -1;
    }
  });

  return arr.map((kv) => kv[0] + ":" + kv[1] + ";").join("");
}

function isLower(letter) {
  return letter >= "a" && letter <= "z";
}

function isUpper(letter) {
  return letter >= "A" && letter <= "Z";
}
```

Python算法源码

```python
# 输入获取
import functools

s = input()


def cmp(a, b):
    if a[1] != b[1]:
        return b[1] - a[1]

    if (a[0].islower() and b[0].islower()) or (a[0].isupper() and b[0].isupper()):
        return 1 if a[0] > b[0] else -1
    else:
        if a[0].isupper():
            return 1
        else:
            return -1


# 算法入口
def getResult():
    letter = {}

    for c in s:
        letter[c] = letter.get(c, 0) + 1

    letterList = list(letter.items())

    letterList.sort(key=functools.cmp_to_key(cmp))

    return "".join(list(map(lambda x: f"{x[0]}:{x[1]};", letterList)))


# 算法调用
print(getResult())
```