题目描述

磁盘的容量单位常用的有M，G，T这三个等级，

它们之间的换算关系为1T = 1024G，1G = 1024M，

现在给定n块磁盘的容量，请对它们按从小到大的顺序进行稳定排序，

例如给定5块盘的容量，1T，20M，3G，10G6T，3M12G9M

排序后的结果为20M，3G，3M12G9M，1T，10G6T。

注意单位可以重复出现，上述3M12G9M表示的容量即为3M+12G+9M，和12M12G相等。

输入描述

输入第一行包含一个整数n(2 <= n <= 100)，表示磁盘的个数。

接下的n行，每行一个字符串(长度大于2，小于30)，

表示磁盘的容量，由一个或多个格式为mv的子串组成，

其中m表示容量大小，v表示容量单位，例如20M，1T，30G，10G6T，3M12G9M。

磁盘容量m的范围为1到1024的正整数，

容量单位v的范围只包含题目中提到的M，G，T三种，换算关系如题目描述。

输出描述

输出n行，表示n块磁盘容量排序后的结果。

用例

输入	3 1G 2G 1024M
输出	1G 1024M 2G
说明	1G和1024M容量相等，稳定排序要求保留它们原来的相对位置，故1G在1024M之前。

输入	3 2G4M 3M2G 1T
输出	3M2G 2G4M 1T
说明	1T的容量大于2G4M，2G4M的容量大于3M2G。

题目解析

简单的字符串操作+排序，具体逻辑请看代码。

Java算法源码

```java
import java.util.Arrays;
import java.util.Scanner;

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

    int n = sc.nextInt();

    String[] capacitys = new String[n];
    for (int i = 0; i < n; i++) capacitys[i] = sc.next();

    getResult(n, capacitys);
  }

  public static void getResult(int n, String[] capacitys) {
    Arrays.sort(capacitys, (a, b) -> calc(a) - calc(b));
    for (String capacity : capacitys) {
      System.out.println(capacity);
    }
  }

  public static int calc(String capacity) {
    int ans = 0;

    StringBuilder num = new StringBuilder();
    for (int i = 0; i < capacity.length(); i++) {
      char c = capacity.charAt(i);

      if (c >= '0' && c <= '9') {
        num.append(c);
      } else {
        switch (c) {
          case 'M':
            ans += Integer.parseInt(num.toString());
            break;
          case 'G':
            ans += Integer.parseInt(num.toString()) * 1024;
            break;
          case 'T':
            ans += Integer.parseInt(num.toString()) * 1024 * 1024;
            break;
        }

        num = new StringBuilder();
      }
    }

    return ans;
  }
}
```

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,
});

const lines = [];
let n;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 1) {
    n = lines[0] - 0;
  }

  if (n && lines.length === n + 1) {
    lines.shift();
    getResult(lines);
    lines.length = 0;
  }
});

function getResult(capacitys) {
  capacitys
    .sort((a, b) => calc(a) - calc(b))
    .forEach((cap) => console.log(cap));
}

function calc(capacity) {
  let ans = 0;

  const num = [];
  for (let i = 0; i < capacity.length; i++) {
    const c = capacity[i];

    if (c >= "0" && c <= "9") {
      num.push(c);
    } else {
      switch (c) {
        case "M":
          ans += parseInt(num.join(""));
          break;
        case "G":
          ans += parseInt(num.join("")) * 1024;
          break;
        case "T":
          ans += parseInt(num.join("")) * 1024 * 1024;
          break;
      }
      num.length = 0;
    }
  }

  return ans;
}
```

Python算法源码

```python
# 输入获取
n = int(input())
capacitys = [input() for _ in range(n)]


def calc(cap):
    ans = 0

    num = []
    for i in range(len(cap)):
        c = cap[i]

        if '0' <= c <= '9':
            num.append(c)
        else:
            if c == 'M':
                ans += int("".join(num))
            elif c == 'G':
                ans += int("".join(num)) * 1024
            elif c == 'T':
                ans += int("".join(num)) * 1024 * 1024

            num = []

    return ans


# 算法入口
def getResult():
    capacitys.sort(key=lambda x: calc(x))

    for cap in capacitys:
        print(cap)


# 算法调用
getResult()
```