1021 个位数统计 (15分)

返回

1021 个位数统计 (15分)

给定一个 k 位整数 $N = d_{k-1}10^{k-1} + \cdots + d_1 10^1 + d_0$ ($0 \le d_i \le 9, i = 0, \cdots, k-1, d_{k-1} > 0$)，请编写程序统计每种不同的个位数字出现的次数。例如：给定 $N = 100311$，则有 2 个 0，3 个 1，和 1 个 3。

输入格式:

每个输入包含 1 个测试用例，即一个不超过 1000 位的正整数 N。

输出格式:

对 N 中每一种不同的个位数字，以 `D:M` 的格式在一行中输出该位数字 `D` 及其在 N 中出现的次数 `M`。要求按 `D` 的升序输出。

输入样例:

```
100311
```

输出样例:

```
0:2
1:3
3:1
```

```cpp
#include <iostream>

using namespace std;

int cnt[10];
int main()
{
    string s;
    cin >> s;
    for(auto c:s)
    {
        int res = c - '0';
        cnt[res] ++;
    }
    for(int i = 0;i <= 9;i ++)
    {
        if(cnt[i] != 0) printf("%d:%d\n",i,cnt[i]);
    }
}
```

1022 D进制的A+B (20分)

1022 D进制的A+B (20分)

输入两个非负 10 进制整数 A 和 B ($\leq 2^{30}-1$)，输出 $A+B$ 的 D ($1 < D \leq 10$)进制数。

输入格式:

输入在一行中依次给出 3 个整数 A、B 和 D。

输出格式:

输出 $A+B$ 的 D 进制数。

输入样例:

```
123 456 8
```

输出样例:

```
1103
```

```cpp
#include <iostream>
using namespace std;
int main() {
    int a, b, d;
    cin >> a >> b >> d;
    int t = a + b;
    if (t == 0) {
        cout << 0;
        return 0;
    }
    int s[100];
    int i = 0;
    while (t != 0) {
        s[i++] = t % d;
        t = t / d;
    }
    for (int j = i - 1; j >= 0; j--)
        cout << s[j];
    return 0;
}
```

1023 组个最小数 (20分)

1023 组个最小数 (20分)

给定数字 0-9 各若干个。你可以以任意顺序排列这些数字，但必须全部使用。目标是使得最后得到的数尽可能小（注意 0 不能做首位）。例如：给定两个 0，两个 1，三个 5，一个 8，我们得到的最小的数就是 10015558。

现给定数字，请编写程序输出能够组成的最小的数。

输入格式:

输入在一行中给出 10 个非负整数，顺序表示我们拥有数字 0、数字 1、……数字 9 的个数。整数间用一个空格分隔。10 个数字的总个数不超过 50，且至少拥有 1 个非 0 的数字。

输出格式:

在一行中输出能够组成的最小的数。

输入样例:

```
2 2 0 0 0 3 0 0 1 0
```

输出样例:

```
10015558
```

```cpp
#include <iostream>
using namespace std;
int main() {
    int a[10], t;
    for (int i = 0; i < 10; i++)
        cin >> a[i];
    for (int i = 1; i < 10; i++) {
        if (a[i] != 0) {
            cout << i;
            t = i;
            break;
        }
    }
    for (int i = 0; i < a[0]; i++) cout << 0;
    for (int i = 0; i < a[t] - 1; i++) cout << t;
    for (int i = t + 1; i < 10; i++)
        for (int k = 0; k < a[i]; k++)
            cout << i;
    return 0;
}
```

1024 科学计数法 (20分)

1024 科学计数法 (20分)

科学计数法是科学家用来表示很大或很小的数字的一种方便的方法，其满足正则表达式 [+-][1-9].[0-9]+E[+-][0-9]+，即数字的整数部分只有 1 位，小数部分至少有 1 位，该数字及其指数部分的正负号即使对正数也必定明确给出。

现以科学计数法的格式给出实数 A，请编写程序按普通数字表示法输出 A，并保证所有有效位都被保留。

输入格式:

每个输入包含 1 个测试用例，即一个以科学计数法表示的实数 A。该数字的存储长度不超过 9999 字节，且其指数的绝对值不超过 9999。

输出格式:

对每个测试用例，在一行中按普通数字表示法输出 A，并保证所有有效位都被保留，包括末尾的 0。

输入样例 1:

```
+1.23400E-03
```

输出样例 1:

```
0.00123400
```

输入样例 2:

```
-1.2E+10
```

输出样例 2:

```
-12000000000
```

```cpp
#include <iostream>
using namespace std;
int main() {
    string s;
    cin >> s;
    int i = 0;
    while (s[i] != 'E') i++;
    string t = s.substr(1, i-1);
    int n = stoi(s.substr(i+1));
    if (s[0] == '-') cout << "-";
    if (n < 0) {
        cout << "0.";
        for (int j = 0; j < abs(n) - 1; j++) cout << '0';
        for (int j = 0; j < t.length(); j++)
            if (t[j] != '.') cout << t[j];
    } else {
        cout << t[0];
        int cnt, j;
```

```
        for (j = 2, cnt = 0; j < t.length() && cnt < n; j++, cnt++) cout <<
t[j];
        if (j == t.length()) {
            for (int k = 0; k < n - cnt; k++) cout << '0';
        } else {
            cout << '.';
            for (int k = j; k < t.length(); k++) cout << t[k];
        }
    }
    return 0;
}
```

1025 反转链表 (25分)

返回

1025 反转链表 (25分)

给定一个常数 K 以及一个单链表 L，请编写程序将 L 中每 K 个结点反转。例如：给定 L 为 1→2→3→4→5→6，K 为 3，则输出应该为 3→2→1→6→5→4；如果 K 为 4，则输出应该为 4→3→2→1→5→6，即最后不到 K 个元素不反转。

输入格式:

每个输入包含 1 个测试用例。每个测试用例第 1 行给出第 1 个结点的地址、结点总个数正整数 N ($\leq 10^5$)、以及正整数 K ($\leq N$)，即要求反转的子链结点的个数。结点的地址是 5 位非负整数，NULL 地址用 -1 表示。

接下来有 N 行，每行格式为：

```
Address Data Next
```

其中 `Address` 是结点地址， `Data` 是该结点保存的整数数据， `Next` 是下一结点的地址。

输出格式:

对每个测试用例，顺序输出反转后的链表，其上每个结点占一行，格式与输入相同。

输入样例：

```
00100 6 4
00000 4 99999
00100 1 12309
68237 6 -1
33218 3 00000
99999 5 68237
12309 2 33218
```

输出样例：

```
00000 4 33218
33218 3 12309
12309 2 00100
00100 1 99999
99999 5 68237
68237 6 -1
```

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    int first, k, n, temp;
    cin >> first >> n >> k;
    int data[100005], next[100005], list[100005];
    for (int i = 0; i < n; i++) {
        cin >> temp;
        cin >> data[temp] >> next[temp];
    }
    int sum = 0;//不一定所有的输入的结点都是有用的，加个计数器
    while (first != -1) {
        list[sum++] = first;
        first = next[first];
    }
    for (int i = 0; i < (sum - sum % k); i += k)
        reverse(begin(list) + i, begin(list) + i + k);
    for (int i = 0; i < sum - 1; i++)
        printf("%05d %d %05d\n", list[i], data[list[i]], list[i + 1]);
    printf("%05d %d -1", list[sum - 1], data[list[sum - 1]]);
    return 0;
}
```

1026 程序运行时间 (15分)

1026 程序运行时间 (15分)

要获得一个 C 语言程序的运行时间，常用的方法是调用头文件 time.h，其中提供了 clock() 函数，可以捕捉从程序开始运行到 clock() 被调用时所耗费的时间。这个时间单位是 clock tick，即“时钟打点”。同时还有一个常数 CLK_TCK，给出了机器时钟每秒所走的时钟打点数。于是为了获得一个函数 f 的运行时间，我们只要在调用 f 之前先调用 clock()，获得一个时钟打点数 C1；在 f 执行完成后再调用 clock()，获得另一个时钟打点数 C2；两次获得的时钟打点数之差 (C2-C1) 就是 f 运行所消耗的时钟打点数，再除以常数 CLK_TCK，就得到了以秒为单位的运行时间。

这里不妨简单假设常数 CLK_TCK 为 100。现给定被测函数前后两次获得的时钟打点数，请你给出被测函数运行的时间。

输入格式:

输入在一行中顺序给出 2 个整数 C1 和 C2。注意两次获得的时钟打点数肯定不相同，即 C1 < C2，并且取值在 $[0, 10^7]$。

输出格式:

在一行中输出被测函数运行的时间。运行时间必须按照 `hh:mm:ss` （即2位的 `时:分:秒`）格式输出；不足 1 秒的时间四舍五入到秒。

输入样例:

```
123 4577973
```

输出样例:

```
12:42:59
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    int c1,c2;
    cin >> c1 >> c2;
    int all_s = ((c2 - c1) + 50)/100;
    int h = all_s / 3600;
    all_s = all_s % 3600;
    int m = all_s / 60;
    int s = all_s % 60;
    printf("%02d:%02d:%02d",h,m,s);
}
```

1027 打印沙漏 (20分)

1027 打印沙漏 (20分)

本题要求你写个程序把给定的符号打印成沙漏的形状。例如给定17个“*”，要求按下列格式打印

```
*****
 ***
  *
 ***
*****
```

所谓“沙漏形状”，是指每行输出奇数个符号；各行符号中心对齐；相邻两行符号数差2；符号数先从大到小顺序递减到1，再从小到大顺序递增；首尾符号数相等。

给定任意N个符号，不一定能正好组成一个沙漏。要求打印出的沙漏能用掉尽可能多的符号。

输入格式:

输入在一行给出1个正整数N（≤1000）和一个符号，中间以空格分隔。

输出格式:

首先打印出由给定符号组成的最大的沙漏形状，最后在一行中输出剩下没用掉的符号数。

输入样例:

```
19 *
```

输出样例:

```
*****
 ***
  *
 ***
*****
2
```

```cpp
#include <iostream>
using namespace std;
int main() {
    int N, row = 0;
    char c;
    cin >> N >> c;
    for (int i = 0; i < N; i++) {
        if ((2 * i * (i + 2) + 1) > N) {
            row = i - 1;
            break;
        }
    }
    for (int i = row; i >= 1; i--) {
        for (int k = row - i; k >= 1; k--) cout << " ";
        for (int j = i * 2 + 1; j >= 1; j--) cout << c;
        cout << endl;
    }
    for (int i = 0; i < row; i++) cout << " ";
    cout << c << endl;
```

```
    for (int i = 1; i <= row; i++) {
        for (int k = row - i; k >= 1; k--) cout << " ";
        for (int j = i * 2 + 1; j >= 1; j--) cout << c;
        cout << endl;
    }
    cout << (N - (2 * row * (row + 2) + 1));
    return 0;
}
```

1028 人口普查 (20分)

‹ 返回

1028 人口普查 (20分)

某城镇进行人口普查，得到了全体居民的生日。现请你写个程序，找出镇上最年长和最年轻的人。

这里确保每个输入的日期都是合法的，但不一定是合理的——假设已知镇上没有超过 200 岁的老人，而今天是 2014 年 9 月 6 日，所以超过 200 岁的生日和未出生的生日都是不合理的，应该被过滤掉。

输入格式:

输入在第一行给出正整数 N，取值在$(0, 10^5]$；随后 N 行，每行给出 1 个人的姓名（由不超过 5 个英文字母组成的字符串）、以及按 `yyyy/mm/dd` （即年/月/日）格式给出的生日。题目保证最年长和最年轻的人没有并列。

输出格式:

在一行中顺序输出有效生日的个数、最年长人和最年轻人的姓名，其间以空格分隔。

输入样例:

```
5
John 2001/05/12
Tom 1814/09/06
Ann 2121/01/30
James 1814/09/05
Steve 1967/11/20
```

输出样例:

```
3 Tom John
```

```
#include <iostream>
using namespace std;
int main() {
    int n, cnt = 0;
    cin >> n;
    string name, birth, maxname, minname, maxbirth = "1814/09/06", minbirth =
"2014/09/06";
    for (int i = 0; i < n; i++) {
        cin >> name >> birth;
        if (birth >= "1814/09/06" && birth <= "2014/09/06") {
            cnt++;
```

```
            if (birth >= maxbirth) {
                maxbirth = birth;
                maxname = name;
            }
            if (birth <= minbirth) {
                minbirth = birth;
                minname = name;
            }
        }
    }
    cout << cnt;
    if (cnt != 0) cout << " " << minname << " " << maxname;
    return 0;
}
```

1029 旧键盘 (20分)

1029 旧键盘 (20分)

旧键盘上坏了几个键，于是在敲一段文字的时候，对应的字符就不会出现。现在给出应该输入的一段文字、以及实际被输入的文字，请你列出肯定坏掉的那些键。

输入格式:

输入在 2 行中分别给出应该输入的文字、以及实际被输入的文字。每段文字是不超过 80 个字符的串，由字母 A-Z（包括大、小写）、数字 0-9、以及下划线 `_`（代表空格）组成。题目保证 2 个字符串均非空。

输出格式:

按照发现顺序，在一行中输出坏掉的键。其中英文字母只输出大写，每个坏键只输出一次。题目保证至少有 1 个坏键。

输入样例:

```
7_This_is_a_test
_hs_s_a_es
```

输出样例:

```
7TI
```

```
#include <iostream>
#include <cctype>
using namespace std;
int main() {
    string s1, s2, ans;
    cin >> s1 >> s2;
    for (int i = 0; i < s1.length(); i++)
        if (s2.find(s1[i]) == string::npos && ans.find(toupper(s1[i])) ==
string::npos)
            ans += toupper(s1[i]);
    cout << ans;
    return 0;
}
```

1030 完美数列 (25分)

1030 完美数列 (25分)

给定一个正整数数列，和正整数 p，设这个数列中的最大值是 M，最小值是 m，如果 $M \leq mp$，则称这个数列是完美数列。

现在给定参数 p 和一些正整数，请你从中选择尽可能多的数构成一个完美数列。

输入格式:

输入第一行给出两个正整数 N 和 p，其中 N（$\leq 10^5$）是输入的正整数的个数，p（$\leq 10^9$）是给定的参数。第二行给出 N 个正整数，每个数不超过 10^9。

输出格式:

在一行中输出最多可以选择多少个数可以用它们组成一个完美数列。

输入样例:

```
10 8
2 3 20 4 5 1 6 7 8 9
```

输出样例:

```
8
```

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main() {
    int n;
    long long p;
    scanf("%d%lld", &n, &p);
    vector<int> v(n);
    for (int i = 0; i < n; i++)
```

```
        cin >> v[i];
    sort(v.begin(), v.end());
    int result = 0, temp = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i + result; j < n; j++) {
            if (v[j] <= v[i] * p) {
                temp = j - i + 1;
                if (temp > result)
                    result = temp;
            } else {
                break;
            }
        }
    }
    cout << result;
    return 0;
}
```

1031 查验身份证 (15分)

返回

1031 查验身份证 (15分)

一个合法的身份证号码由17位地区、日期编号和顺序编号加1位校验码组成。校验码的计算规则如下：

首先对前17位数字加权求和，权重分配为：{7，9，10，5，8，4，2，1，6，3，7，9，10，5，8，4，2}；然后将计算的和对11取模得到值 `Z`；最后按照以下关系对应 `Z` 值与校验码 `M` 的值：

```
Z: 0 1 2 3 4 5 6 7 8 9 10
M: 1 0 X 9 8 7 6 5 4 3 2
```

现在给定一些身份证号码，请你验证校验码的有效性，并输出有问题的号码。

输入格式：

输入第一行给出正整数N（≤ 100）是输入的身份证号码的个数。随后N行，每行给出1个18位身份证号码。

输出格式：

按照输入的顺序每行输出1个有问题的身份证号码。这里并不检验前17位是否合理，只检查前17位是否全为数字且最后1位校验码计算准确。如果所有号码都正常，则输出 `All passed`。

```
// #include <iostream>
// #include <vector>

// using namespace std;

// int W[] = {7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2};
// char M[] = "10X98765432";


// int main()
// {
//     int n;
```

```
//     int ans = 0;
//     cin >> n;
//     vector<string> res;
//     while(n --)
//     {
//         string s;
//         cin >> s;
//         for(int i = 0;i <= 16;i ++) ans += W[i] * (s[i] - '0');
//         ans = ans % 11;
//         if(M[ans] != s[17]) res.push_back(s);
//     }


//     if(res.size())
//         for(int i = 0;i < res.size();i ++) cout << res[i] <<endl;
//     else cout << "All passed";
// }
#include <iostream>
using namespace std;
int a[17] = {7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2};
int b[11] = {1, 0, 10, 9, 8, 7, 6, 5, 4, 3, 2};
string s;
bool isTrue() {
    int sum = 0;
    for (int i = 0; i < 17; i++) {
        if (s[i] < '0' || s[i] > '9') return false;
        sum += (s[i] - '0') * a[i];
    }
    int temp = (s[17] == 'X') ? 10 : (s[17] - '0');
    return b[sum%11] == temp;
}
int main() {
    int n, flag = 0;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> s;
        if (!isTrue()) {
            cout << s << endl;
            flag = 1;
        }
    }
    if (flag == 0) cout << "All passed";
    return 0;
}
```

1032 挖掘机技术哪家强 (20分)

1032 挖掘机技术哪家强 (20分)

为了用事实说明挖掘机技术到底哪家强，PAT 组织了一场挖掘机技能大赛。现请你根据比赛结果统计出技术最强的那个学校。

输入格式:

输入在第 1 行给出不超过 10^5 的正整数 N，即参赛人数。随后 N 行，每行给出一位参赛者的信息和成绩，包括其所代表的学校的编号（从 1 开始连续编号）、及其比赛成绩（百分制），中间以空格分隔。

输出格式:

在一行中给出总得分最高的学校的编号、及其总分，中间以空格分隔。题目保证答案唯一，没有并列。

输入样例:

```
6
3 65
2 80
1 100
2 70
3 40
3 0
```

输出样例:

```
2 150
```

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int N;
    cin >> N;
    vector<int> a(N + 1);
    int num, score;
    for (int i = 0; i < N; i++) {
        cin >> num >> score;
        a[num] += score;
    }
    int max = a[1], t = 1;
    for (int i = 2; i <= N; i++) {
        if (max < a[i]) {
            max = a[i];
            t = i;
        }
    }
    cout << t << " " << max;
    return 0;
}
```

1033 旧键盘打字 (20分)

1033 旧键盘打字 (20分)

旧键盘上坏了几个键，于是在敲一段文字的时候，对应的字符就不会出现。现在给出应该输入的一段文字、以及坏掉的那些键，打出的结果文字会是怎样？

输入格式:

输入在 2 行中分别给出坏掉的那些键、以及应该输入的文字。其中对应英文字母的坏键以大写给出；每段文字是不超过 10^5 个字符的串。可用的字符包括字母 [`a`-`z`, `A`-`Z`]、数字 `0`-`9`、以及下划线 `_`（代表空格）、`,`、`.`、`-`、`+`（代表上档键）。题目保证第 2 行输入的文字串非空。

注意：如果上档键坏掉了，那么大写的英文字母无法被打出。

输出格式:

在一行中输出能够被打出的结果文字。如果没有一个字符能被打出，则输出空行。

输入样例:

```
7+IE.
7_This_is_a_test.
```

输出样例:

```
_hs_s_a_tst
```

```
#include <iostream>
#include <cctype>
using namespace std;
int main() {
  string bad, should;
  getline(cin, bad);
  getline(cin, should);
  for (int i = 0, length = should.length(); i < length; i++) {
    if (bad.find(toupper(should[i])) != string::npos) continue;
    if (isupper(should[i]) && bad.find('+') != string::npos) continue;
    cout << should[i];
  }
  return 0;
}
```

1034 有理数四则运算 (20分)

1034 有理数四则运算 (20分)

本题要求编写程序，计算 2 个有理数的和、差、积、商。

输入格式:

输入在一行中按照 `a1/b1 a2/b2` 的格式给出两个分数形式的有理数，其中分子和分母全是整型范围内的整数，负号只可能出现在分子前，分母不为 0。

输出格式:

分别在 4 行中按照 `有理数1 运算符 有理数2 = 结果` 的格式顺序输出 2 个有理数的和、差、积、商。注意输出的每个有理数必须是该有理数的最简形式 `k a/b`，其中 `k` 是整数部分，`a/b` 是最简分数部分；若为负数，则须加括号；若除法分母为 0，则输出 `Inf`。题目保证正确的输出中没有超过整型范围的整数。

输入样例 1:

```
2/3 -4/2
```

输出样例 1:

```
2/3 + (-2) = (-1 1/3)
2/3 - (-2) = 2 2/3
2/3 * (-2) = (-1 1/3)
2/3 / (-2) = (-1/3)
```

```cpp
#include <iostream>
#include <cmath>
using namespace std;
long long a, b, c, d;
long long gcd(long long t1, long long t2) {
    return t2 == 0 ? t1 : gcd(t2, t1 % t2);
}
void func(long long m, long long n) {
    if (m * n == 0) {
        printf("%s", n == 0 ? "Inf" : "0");
        return ;
    }
    bool flag = ((m < 0 && n > 0) || (m > 0 && n < 0));
    m = abs(m); n = abs(n);
    long long x = m / n;
    printf("%s", flag ? "(-" : "");
    if (x != 0) printf("%lld", x);
    if (m % n == 0) {
        if(flag) printf(")");
        return ;
    }
    if (x != 0) printf(" ");
    m = m - x * n;
    long long t = gcd(m, n);
    m = m / t; n = n / t;
    printf("%lld/%lld%s", m, n, flag ? ")" : "");
```

```
}
int main() {
    scanf("%lld/%lld %lld/%lld", &a, &b, &c, &d);
    func(a, b); printf(" + "); func(c, d); printf(" = "); func(a * d + b * c, b
* d); printf("\n");
    func(a, b); printf(" - "); func(c, d); printf(" = "); func(a * d - b * c, b
* d); printf("\n");
    func(a, b); printf(" * "); func(c, d); printf(" = "); func(a * c, b * d);
printf("\n");
    func(a, b); printf(" / "); func(c, d); printf(" = "); func(a * d, b * c);
    return 0;
}
```

1035 插入与归并 (25分)

1035 插入与归并 (25分)

根据维基百科的定义：

插入排序是迭代算法，逐一获得输入数据，逐步产生有序的输出序列。每步迭代中，算法从输入序列中取出一元素，将之插入有序序列中正确的位置。如此迭代直到全部元素有序。

归并排序进行如下迭代操作：首先将原始序列看成 N 个只包含 1 个元素的有序子序列，然后每次迭代归并两个相邻的有序子序列，直到最后只剩下 1 个有序的序列。

现给定原始序列和由某排序算法产生的中间序列，请你判断该算法究竟是哪种排序算法？

输入格式:

输入在第一行给出正整数 N (≤100)；随后一行给出原始序列的 N 个整数；最后一行给出由某排序算法产生的中间序列。这里假设排序的目标序列是升序。数字间以空格分隔。

输出格式:

首先在第 1 行中输出 `Insertion Sort` 表示插入排序、或 `Merge Sort` 表示归并排序；然后在第 2 行中输出用该排序算法再迭代一轮的结果序列。题目保证每组测试的结果是唯一的。数字间以空格分隔，且行首尾不得有多余空格。

输入样例 1:

```
10
3 1 2 8 7 5 9 4 6 0
1 2 3 7 8 5 9 4 6 0
```

输出样例 1:

```
Insertion Sort
1 2 3 5 7 8 9 4 6 0
```

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    int n, a[100], b[100], i, j;
    cin >> n;
```

```
    for (int i = 0; i < n; i++)
        cin >> a[i];
    for (int i = 0; i < n; i++)
        cin >> b[i];
    for (i = 0; i < n - 1 && b[i] <= b[i + 1]; i++);
    for (j = i + 1; a[j] == b[j] && j < n; j++);
    if (j == n) {
        cout << "Insertion Sort" << endl;
        sort(a, a + i + 2);
    } else {
        cout << "Merge Sort" << endl;
        int k = 1, flag = 1;
        while(flag) {
            flag = 0;
            for (i = 0; i < n; i++) {
                if (a[i] != b[i])
                    flag = 1;
            }
            k = k * 2;
            for (i = 0; i < n / k; i++)
                sort(a + i * k, a + (i + 1) * k);
            sort(a + n / k * k, a + n);
        }
    }
    for (j = 0; j < n; j++) {
        if (j != 0) printf(" ");
        printf("%d", a[j]);
    }
    return 0;
}
```

1036 跟奥巴马一起编程 (15分)

1036 跟奥巴马一起编程 (15分)

美国总统奥巴马不仅呼吁所有人都学习编程，甚至以身作则编写代码，成为美国历史上首位编写计算机代码的总统。2014 年底，为庆祝“计算机科学教育周”正式启动，奥巴马编写了很简单的计算机代码：在屏幕上画一个正方形。现在你也跟他一起画吧！

输入格式:

输入在一行中给出正方形边长 N（$3 \le N \le 20$）和组成正方形边的某种字符 C，间隔一个空格。

输出格式:

输出由给定字符 C 画出的正方形。但是注意到行间距比列间距大，所以为了让结果看上去更像正方形，我们输出的行数实际上是列数的 50%（四舍五入取整）。

输入样例:

```
10 a
```

输出样例:

```
aaaaaaaaaa
a        a
a        a
a        a
aaaaaaaaaa
```

```cpp
#include <iostream>
using namespace std;
int main() {
    int N;
    char c;
    cin >> N >> c;
    int t = N / 2 + N % 2;
    for (int i = 0; i < N; i++)
        cout << c;
    cout << endl;
    for (int i = 0; i < t - 2; i++) {
        cout << c;
        for (int k = 0; k < N - 2; k++)
            cout << " ";
        cout << c << endl;
    }
    for (int i = 0; i < N; i++)
        cout << c;
    return 0;
}
```

1037 在霍格沃茨找零钱 (20分)

1037 在霍格沃茨找零钱 (20分)

如果你是哈利·波特迷，你会知道魔法世界有它自己的货币系统 —— 就如海格告诉哈利的：“十七个银西可(Sickle)兑一个加隆(Galleon)，二十九个纳特(Knut)兑一个西可，很容易。”现在，给定哈利应付的价钱 P 和他实付的钱 A，你的任务是写一个程序来计算他应该被找的零钱。

输入格式:

输入在 1 行中分别给出 P 和 A，格式为 `Galleon.Sickle.Knut`，其间用 1 个空格分隔。这里 `Galleon` 是 [0, 10^7] 区间内的整数，`Sickle` 是 [0, 17) 区间内的整数，`Knut` 是 [0, 29) 区间内的整数。

输出格式:

在一行中用与输入同样的格式输出哈利应该被找的零钱。如果他没带够钱，那么输出的应该是负数。

输入样例 1:

```
10.16.27 14.1.28
```

输出样例 1:

```
3.2.1
```

输入样例 2:

```
14.1.28 10.16.27
```

输出样例 2:

```
-3.2.1
```

```cpp
#include <iostream>

using namespace std;

int main() {
    int a, b ,c, m, n, t, x, y, z;
    scanf("%d.%d.%d %d.%d.%d",&a, &b, &c, &m, &n, &t);

    if (a > m || (a == m && b > n) || (a == m && b == n && c > t)) {
        swap(a, m); swap(b, n); swap(c, t);
        printf("-");
    }//the large one
    z = t < c ? t - c + 29 : t - c;
    n = t < c ? n - 1 : n;
    y = n < b ? n - b + 17 : n - b;
    x = n < b ? m - a - 1 : m - a;
    printf("%d.%d.%d", x, y, z);
    return 0;
}
```

1038 统计同成绩学生 (20分)

1038 统计同成绩学生 (20分)

本题要求读入 N 名学生的成绩，将获得某一给定分数的学生人数输出。

输入格式:

输入在第 1 行给出不超过 10^5 的正整数 N，即学生总人数。随后一行给出 N 名学生的百分制整数成绩，中间以空格分隔。最后一行给出要查询的分数个数 K（不超过 N 的正整数），随后是 K 个分数，中间以空格分隔。

输出格式:

在一行中按查询顺序给出得分等于指定分数的学生人数，中间以空格分隔，但行末不得有多余空格。

输入样例:

```
10
60 75 90 55 75 99 82 90 75 50
3 75 90 88
```

输出样例:

```
3 2 0
```

```
// #include <iostream>
// #include <set>

// using namespace std;

// int main()
// {
//     int n;
//     cin >> n;
//     multiset<int> hash;
//     for(int i = 0;i < n;i ++)
//     {
//         int score;
//         cin >> score;
//         hash.insert(score);
//     }
//     int T;
//     cin >>T;
//     while(T--)
//     {
//         int Score;
//         cin >> Score;
//         cout << hash.count(Score)<<' ';
//     }

// }
```

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, m, temp;
    scanf("%d", &n);
    vector<int> b(101);
    for (int i = 0; i < n; i++) {
        scanf("%d", &temp);
        b[temp]++;
    }
    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d", &temp);
        if (i != 0) printf(" ");
        printf("%d", b[temp]);
    }
    return 0;
}
```

1039 到底买不买 (20分)

返回

1039 到底买不买 (20分)

小红想买些珠子做一串自己喜欢的珠串。卖珠子的摊主有很多串五颜六色的珠串，但是不肯把任何一串拆散了卖。于是小红要你帮忙判断一下，某串珠子里是否包含了全部自己想要的珠子？如果是，那么告诉她有多少多余的珠子；如果不是，那么告诉她缺了多少珠子。

为方便起见，我们用[0-9]、[a-z]、[A-Z]范围内的字符来表示颜色。例如在图1中，第3串是小红想做的珠串；那么第1串可以买，因为包含了全部她想要的珠子，还多了8颗不需要的珠子；第2串不能买，因为没有黑色珠子，并且少了一颗红色的珠子。

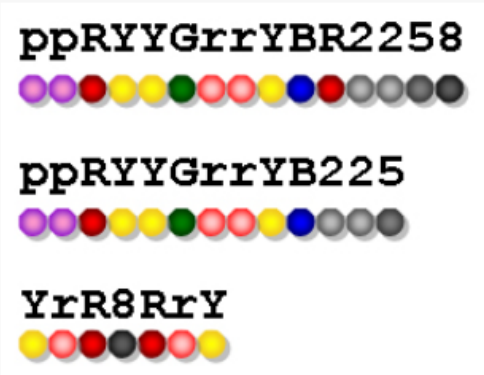

图 1

输入格式:

每个输入包含 1 个测试用例。每个测试用例分别在 2 行中先后给出摊主的珠串和小红想做的珠串，两串都不超过 1000 个珠子。

输出格式:

如果可以买，则在一行中输出 `Yes` 以及有多少多余的珠子；如果不可以买，则在一行中输出 `No` 以及缺了多少珠子。其间以 1 个空格分隔。

输入样例 1:

```
ppRYYGrrYBR2258
YrR8RrY
```

输出样例 1:

```
Yes 8
```

输入样例 2:

```
ppRYYGrrYB225
YrR8RrY
```

输出样例 2:

```
No 2
```

```cpp
#include <iostream>

using namespace std;

int book[256];

int main() {
    string a, b;
    cin >> a >> b;
    for (int i = 0; i < a.size(); i++)
        book[a[i]]++;
    int result = 0;
    for (int i = 0; i < b.size(); i++) {
        if (book[b[i]] > 0)
            book[b[i]]--;
        else
            result++;
    }
    if(result != 0)
        printf("No %d", result);
    else
        printf("Yes %d", a.size() - b.size());
    return 0;
}
```

1040 有几个PAT (25分)

1040 有几个PAT (25分)

字符串 APPAPT 中包含了两个单词 PAT，其中第一个 PAT 是第 2 位(P)，第 4 位(A)，第 6 位(T)；第二个 PAT 是第 3 位(P)，第 4 位(A)，第 6 位(T)。

现给定字符串，问一共可以形成多少个 PAT？

输入格式:

输入只有一行，包含一个字符串，长度不超过10^5，只包含 P、A、T 三种字母。

输出格式:

在一行中输出给定字符串中包含多少个 PAT。由于结果可能比较大，只输出对 1000000007 取余数的结果。

输入样例:

```
APPAPT
```

输出样例:

```
2
```

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    cin >> s;
    int len = s.length(), result = 0, countp = 0, countt = 0;
    for (int i = 0; i < len; i++) {
        if (s[i] == 'T')
            countt++;
    }
    for (int i = 0; i < len; i++) {
        if (s[i] == 'P') countp++;
        if (s[i] == 'T') countt--;
        if (s[i] == 'A') result = (result + (countp * countt) % 1000000007) %
1000000007;
    }
    cout << result;
    return 0;
}
```