1001 害死人不偿命的(3n+1)猜想 (15分)

1001 害死人不偿命的(3n+1)猜想 (15分)

卡拉兹(Callatz)猜想：

对任何一个正整数 n，如果它是偶数，那么把它砍掉一半；如果它是奇数，那么把 $(3n+1)$ 砍掉一半。这样一直反复砍下去，最后一定在某一步得到 $n=1$。卡拉兹在 1950 年的世界数学家大会上公布了这个猜想，传说当时耶鲁大学师生齐动员，拼命想证明这个貌似很傻很天真的命题，结果闹得学生们无心学业，一心只证 $(3n+1)$，以至于有人说这是一个阴谋，卡拉兹是在蓄意延缓美国数学界教学与科研的进展……

我们今天的题目不是证明卡拉兹猜想，而是对给定的任一不超过 1000 的正整数 n，简单地数一下，需要多少步（砍几下）才能得到 $n=1$？

输入格式：

每个测试输入包含 1 个测试用例，即给出正整数 n 的值。

输出格式：

输出从 n 计算到 1 需要的步数。

输入样例：

```
3
```

输出样例：

```
5
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    int n;
    int res;
    cin >> n;
    while(n != 1)
    {
        if(n % 2 == 0) n /= 2;
        else n = (3 * n + 1) / 2;
        res ++;
    }
    cout << res;
}
```

1002 写出这个数 (20分)

1002 写出这个数 (20分)

读入一个正整数 n，计算其各位数字之和，用汉语拼音写出和的每一位数字。

输入格式:

每个测试输入包含 1 个测试用例，即给出自然数 n 的值。这里保证 n 小于 10^{100}。

输出格式:

在一行内输出 n 的各位数字之和的每一位，拼音数字间有 1 空格，但一行中最后一个拼音数字后没有空格。

输入样例:

```
1234567890987654321123456789
```

输出样例:

```
yi san wu
```

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

using namespace std;

string name[] = {"ling","yi","er","san","si","wu","liu","qi","ba","jiu"};
vector<string> ans;
int main()
{
  char c;
  int sum = 0;
  while((c = getchar()) != '\n')
  {
    sum += c-'0';
  }
  if(!sum)  ans.push_back( name[0] );
  while(sum)
  {
    ans.push_back( name[ sum%10 ] );
    sum /= 10;
  }
  cout << ans[ans.size()-1];
  for(int i = ans.size()-2; i >= 0; i--)
    cout << " " << ans[i];
  cout << endl;
  return 0;
}
```

1003 我要通过！ (20分)

1003 我要通过！ (20分)

“**答案正确**”是自动判题系统给出的最令人欢喜的回复。本题属于 PAT 的“**答案正确**”大派送 —— 只要读入的字符串满足下列条件，系统就输出“**答案正确**”，否则输出“**答案错误**”。

得到“**答案正确**”的条件是：

1. 字符串中必须仅有 `P`、 `A`、 `T` 这三种字符，不可以包含其它字符；
2. 任意形如 `xPATx` 的字符串都可以获得“**答案正确**”，其中 `x` 或者是空字符串，或者是仅由字母 `A` 组成的字符串；
3. 如果 `aPbTc` 是正确的，那么 `aPbATca` 也是正确的，其中 `a`、 `b`、 `c` 均或者是空字符串，或者是仅由字母 `A` 组成的字符串。

现在就请你为 PAT 写一个自动裁判程序，判定哪些字符串是可以获得“**答案正确**”的。

输入格式：

每个测试输入包含 1 个测试用例。第 1 行给出一个正整数 n (< 10)，是需要检测的字符串个数。接下来每个字符串占一行，字符串长度不超过 100，且不包含空格。

输出格式：

每个字符串的检测结果占一行，如果该字符串可以获得“**答案正确**”，则输出 `YES`，否则输出 `NO`。

输入样例：

```
8
PAT
PAAT
AAPATAA
AAPAATAAAA
xPATx
PT
Whatever
APAAATAA
```

输出样例：

```
YES
YES
YES
YES
NO
NO
NO
NO
```

```
#include <iostream>
#include <cstring>
#include <vector>
#include <deque>
```

```
#include <map>
#include <string>
#include <algorithm>
using namespace std;



int main()
{
  int n ;
  cin >> n;
  while(n--)
  {
    string str;
    int p = 0, t = 0;
    cin >> str;
    int len = str.size();
    int ok = 1,i,post,posp;
    for(i = 0; i < len; i++)
    {
      if(!(str[i]== 'P' || str[i] == 'A' || str[i] == 'T'))
        ok = 0;
      if(str[i] == 'P')
      {
        p++;
        posp = i;
      }
      if(str[i] == 'T')
      {
        t++;
        post = i;
      }
    }
    if(!ok)  cout << "NO" << endl;
    else
    {
      int a = posp - 0;
      int c = len - post-1;
      int b = post-posp-1;
      //cout << a << " " << b << " " << c << endl;
      if(c==0&&b>=1&&a==0)
        printf("YES\n");
      else if(c&&c==a*b)
        printf("YES\n");
      else
        printf("NO\n");
    }
  }
  return 0;
}
```

1004 成绩排名 (20分)

1004 成绩排名 (20分)

读入 n (>0) 名学生的姓名、学号、成绩，分别输出成绩最高和成绩最低学生的姓名和学号。

输入格式:

每个测试输入包含 1 个测试用例，格式为

```
第 1 行：正整数 n
第 2 行：第 1 个学生的姓名 学号 成绩
第 3 行：第 2 个学生的姓名 学号 成绩
  ... ... ...
第 n+1 行：第 n 个学生的姓名 学号 成绩
```

其中 `姓名` 和 `学号` 均为不超过 10 个字符的字符串，成绩为 0 到 100 之间的一个整数，这里保证在一组测试用例中没有两个学生的成绩是相同的。

输出格式:

对每个测试用例输出 2 行，第 1 行是成绩最高学生的姓名和学号，第 2 行是成绩最低学生的姓名和学号，字符串间有 1 空格。

输入样例:

```
3
Joe Math990112 89
Mike CS991301 100
Mary EE990830 95
```

输出样例:

```
Mike CS991301
Joe Math990112
```

```
#include <iostream>

using namespace std;

struct student
{
    char name[20];
    char date[20];
    int sorce;
};
int main()
{
    int n;
    cin >> n;
    struct student s[1000];
    for(int i = 1;i <= n;i ++) scanf("%s %s
%d",s[i].name,s[i].date,&s[i].sorce);
    int max = -1,min = 101;
    int max_i,min_i;
    for(int i = 1;i <= n;i ++)
    {
```

```
        if(s[i].sorce > max) max_i = i,max = s[i].sorce;
        if(s[i].sorce < min) min_i = i,min = s[i].sorce;
    }
    cout << s[max_i].name << ' ' << s[max_i].date << endl;
    cout << s[min_i].name << ' ' << s[min_i].date << endl;
}
```

1005 继续(3n+1)猜想 (25分)

返回

1005 继续(3n+1)猜想 (25分)

卡拉兹(Callatz)猜想已经在1001中给出了描述。在这个题目里，情况稍微有些复杂。

当我们验证卡拉兹猜想的时候，为了避免重复计算，可以记录下递推过程中遇到的每一个数。例如对 $n = 3$ 进行验证的时候，我们需要计算 3、5、8、4、2、1，则当我们对 $n = 5$、8、4、2 进行验证的时候，就可以直接判定卡拉兹猜想的真伪，而不需要重复计算，因为这 4 个数已经在验证3的时候遇到过了，我们称 5、8、4、2 是被 3 “覆盖”的数。我们称一个数列中的某个数 n 为“关键数”，如果 n 不能被数列中的其他数字所覆盖。

现在给定一系列待验证的数字，我们只需要验证其中的几个关键数，就可以不必再重复验证余下的数字。你的任务就是找出这些关键数字，并按从大到小的顺序输出它们。

输入格式:

每个测试输入包含 1 个测试用例，第 1 行给出一个正整数 K (< 100)，第 2 行给出 K 个互不相同的待验证的正整数 n $(1 < n \leq 100)$的值，数字间用空格隔开。

输出格式:

每个测试用例的输出占一行，按从大到小的顺序输出关键数字。数字间用 1 个空格隔开，但一行中最后一个数字后没有空格。

输入样例:

```
6
3 5 6 7 8 11
```

输出样例:

```
7 6
```

```
#include <iostream>
#include <vector>
#include <cstring>
#include <algorithm>
#include <functional>
using namespace std;

int f[100000];
vector<int> ee;
vector<int> ans;
int main()
{
```

```
    int n,i,e;
    memset(f, 0, sizeof(f));
    cin >> n;
    for(i = 1; i <= n; i++)
    {
        cin >> e;
        ee.push_back(e);
    }
    for(i = 0; i < ee.size(); i++)
    {
        int  tmp = ee[i];
        while(tmp != 1)
        {
            if(tmp & 1)
            {
                tmp = (3*tmp+1) >> 1;
            }
            else
            {
                tmp >>= 1;
            }
            f[tmp] = 1;
        }
    }
    for(i = 0; i < ee.size(); i++)
    {
        if(!f[ee[i]])   ans.push_back(ee[i]);
    }
    sort(ans.begin(), ans.end(), greater<int>());
    cout << ans[0];
    for(i = 1; i < ans.size(); i++)
    {
        cout << " " << ans[i];
    }
    cout << endl;
}
```

1006 换个格式输出整数 (15分)

1006 换个格式输出整数 (15分)

让我们用字母 `B` 来表示“百”、字母 `S` 表示“十”，用 `12...n` 来表示不为零的个位数字 `n`（< 10），换个格式来输出任一个不超过 3 位的正整数。例如 `234` 应该被输出为 `BBSSS1234`，因为它有 2 个“百”、3 个“十”、以及个位的 4。

输入格式:

每个测试输入包含 1 个测试用例，给出正整数 n（< 1000）。

输出格式:

每个测试用例的输出占一行，用规定的格式输出 n。

输入样例 1:

```
234
```

输出样例 1:

```
BBSSS1234
```

输入样例 2:

```
23
```

输出样例 2:

```
SS123
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    int m;
    int b,s;
    cin >> m;
    if(m >= 100)
    {
        if(m / 100 != 0) b = m / 100,m %= 100;
        for(int i = 1;i <= b;i ++) cout << 'B';
    }
    if(m >= 10)
    {
        if(m / 10 != 0 ) s = m / 10,m %= 10;
        for(int i = 1;i <= s;i ++) cout << 'S';
    }
    for(int i = 1;i <= m;i ++) cout << i ;
```

```
}
```

1007 素数对猜想 (20分)

返回

1007 素数对猜想 (20分)

让我们定义d_n为：$d_n = p_{n+1} - p_n$，其中p_i是第i个素数。显然有$d_1 = 1$，且对于$n > 1$有d_n是偶数。“素数对猜想”认为“存在无穷多对相邻且差为2的素数”。

现给定任意正整数 `N` ($< 10^5$)，请计算不超过 `N` 的满足猜想的素数对的个数。

输入格式:

输入在一行给出正整数 `N`。

输出格式:

在一行中输出不超过 `N` 的满足猜想的素数对的个数。

输入样例:

```
20
```

输出样例:

```
4
```

```
#include <iostream>

using namespace std;

bool isPrim(int n)
{
    for(int i = 2; i * i <= n;i ++)
        if(n % i == 0) return false;
    return true;
}
int main()
{
    int N,cnt;
    cin >> N;
    for(int i = 5;i <= N;i ++)
        if(isPrim(i - 2) && isPrim(i)) cnt ++;
    cout << cnt;
}
```

1008 数组元素循环右移问题 (20分)

1008 数组元素循环右移问题 (20分)

一个数组A中存有N（>0）个整数，在不允许使用另外数组的前提下，将每个整数循环向右移M（≥ 0）个位置，即将A中的数据由（$A_0A_1\cdots A_{N-1}$）变换为（$A_{N-M}\cdots A_{N-1}A_0A_1\cdots A_{N-M-1}$）（最后$M$个数循环移至最前面的$M$个位置）。如果需要考虑程序移动数据的次数尽量少，要如何设计移动的方法？

输入格式:

每个输入包含一个测试用例，第1行输入N（$1\leq N\leq 100$）和M（≥ 0）；第2行输入N个整数，之间用空格分隔。

输出格式:

在一行中输出循环右移M位以后的整数序列，之间用空格分隔，序列结尾不能有多余空格。

输入样例:

```
6 2
1 2 3 4 5 6
```

输出样例:

```
5 6 1 2 3 4
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main() {
    int n, m;
    cin >> n >> m;
    vector<int> a(n);
    for (int i = 0; i < n; i++)
        cin >> a[i];
    m %= n;
    if (m != 0) {
        reverse(begin(a), begin(a) + n);
        reverse(begin(a), begin(a) + m);
        reverse(begin(a) + m, begin(a) + n);
    }
    for (int i = 0; i < n - 1; i++)
        cout << a[i] << " ";
    cout << a[n - 1];
    return 0;
}
```

1009 说反话 (20分)

1009 说反话 (20分)

给定一句英语，要求你编写程序，将句中所有单词的顺序颠倒输出。

输入格式:

测试输入包含一个测试用例，在一行内给出总长度不超过 80 的字符串。字符串由若干单词和若干空格组成，其中单词是由英文字母（大小写有区分）组成的字符串，单词之间用 1 个空格分开，输入保证句子末尾没有多余的空格。

输出格式:

每个测试用例的输出占一行，输出倒序后的句子。

输入样例:

```
Hello World Here I Come
```

输出样例:

```
Come I Here World Hello
```

```cpp
#include <iostream>
#include <vector>
using namespace std;

vector<string> res;
int main()
{
    string s;
    while(cin >> s) res.push_back(s);
    for(int i = res.size() - 1;i > 0 ;i --) cout << res[i] << ' ';
    cout << res[0] ;
}
```

1010 一元多项式求导 (25分)

1010 一元多项式求导 (25分)

设计函数求一元多项式的导数。（注：x^n（n为整数）的一阶导数为nx^{n-1}。）

输入格式:

以指数递降方式输入多项式非零项系数和指数（绝对值均为不超过 1000 的整数）。数字间以空格分隔。

输出格式:

以与输入相同的格式输出导数多项式非零项的系数和指数。数字间以空格分隔，但结尾不能有多余空格。注意“零多项式”的指数和系数都是 0，但是表示为 `0 0`。

输入样例:

```
3 4 -5 2 6 1 -2 0
```

输出样例:

```
12 3 -10 1 6 0
```

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

// using namespace std;

// int main()
// {
//     vector<int> a;
//     int m,n;
//     while(cin >> m >> n)
//     {
//         if(n == 0) continue;
//         a.push_back(m * n);
//         a.push_back(n - 1);
//     }
//     for(int i = 0;i < a.size() - 1;i ++) cout << a[i] << ' ';
//     cout << a[a.size()];

// }
#include <iostream>
using namespace std;
int main() {
    int a, b, flag = 0;
    while (cin >> a >> b) {
        if (b != 0) {
            if (flag == 1) cout << " ";
            cout << a * b << " " << b - 1;
            flag = 1;
        }
    }
```

```
    if (flag == 0) cout << "0 0";
    return 0;
}
```

1011 A+B 和 C (15分)

1011 A+B 和 C (15分)

给定区间 $[-2^{31}, 2^{31}]$ 内的 3 个整数 A、B 和 C，请判断 $A+B$ 是否大于 C。

输入格式:

输入第 1 行给出正整数 T (≤ 10)，是测试用例的个数。随后给出 T 组测试用例，每组占一行，顺序给出 A、B 和 C。整数间以空格分隔。

输出格式:

对每组测试用例，在一行中输出 `Case #X: true` 如果 $A+B>C$，否则输出 `Case #X: false`，其中 `X` 是测试用例的编号（从 1 开始）。

输入样例:

```
4
1 2 3
2 3 4
2147483647 0 2147483646
0 -2147483648 -2147483647
```

输出样例:

```
Case #1: false
Case #2: true
Case #3: true
Case #4: false
```

```
#include <iostream>

using namespace std;

int main()
{
    int n;
    cin >> n;
    int res = n;
    while(n --)
    {
        long long a,b,c;
        cin >> a >> b >> c;
        if(a + b > c) cout << "Case #"<< res - n<<": true" << endl;
        else cout <<"Case #"<< res - n <<": false" << endl;
    }
}
```

1012 数字分类 (20分)

1012 数字分类 (20分)

给定一系列正整数，请按要求对数字进行分类，并输出以下 5 个数字：

- A_1 = 能被 5 整除的数字中所有偶数的和；
- A_2 = 将被 5 除后余 1 的数字按给出顺序进行交错求和，即计算 $n_1 - n_2 + n_3 - n_4 \cdots$；
- A_3 = 被 5 除后余 2 的数字的个数；
- A_4 = 被 5 除后余 3 的数字的平均数，精确到小数点后 1 位；
- A_5 = 被 5 除后余 4 的数字中最大数字。

输入格式:

每个输入包含 1 个测试用例。每个测试用例先给出一个不超过 1000 的正整数 N，随后给出 N 个不超过 1000 的待分类的正整数。数字间以空格分隔。

输出格式:

对给定的 N 个正整数，按题目要求计算 $A_1 \sim A_5$ 并在一行中顺序输出。数字间以空格分隔，但行末不得有多余空格。

若其中某一类数字不存在，则在相应位置输出 `N`。

输入样例 1:

```
13 1 2 3 4 5 6 7 8 9 10 20 16 18
```

输出样例 1:

```
30 11 2 9.7 9
```

输入样例 2:

```
8 1 2 4 5 6 7 9 16
```

输出样例 2:

```
N 11 2 N 9
```

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, num, A1 = 0, A2 = 0, A5 = 0;
    double A4 = 0.0;
    cin >> n;
    vector<int> v[5];
    for (int i = 0; i < n; i++) {
```

```
        cin >> num;
        v[num%5].push_back(num);
    }
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < v[i].size(); j++) {
            if (i == 0 && v[i][j] % 2 == 0) A1 += v[i][j];
            if (i == 1 && j % 2 == 0) A2 += v[i][j];
            if (i == 1 && j % 2 == 1) A2 -= v[i][j];
            if (i == 3) A4 += v[i][j];
            if (i == 4 && v[i][j] > A5) A5 = v[i][j];
        }
    }
    for (int i = 0; i < 5; i++) {
        if (i != 0) printf(" ");
        if (i == 0 && A1 == 0 || i != 0 && v[i].size() == 0) {
            printf("N"); continue;
        }
        if (i == 0) printf("%d", A1);
        if (i == 1) printf("%d", A2);
        if (i == 2) printf("%d", v[2].size());
        if (i == 3) printf("%.1f", A4 / v[3].size());
        if (i == 4) printf("%d", A5);
    }
    return 0;
}
```

1013 数素数 (20分)

1013 数素数 (20分)

令 P_i 表示第 i 个素数。现任给两个正整数 $M \le N \le 10^4$，请输出 P_M 到 P_N 的所有素数。

输入格式:

输入在一行中给出 M 和 N，其间以空格分隔。

输出格式:

输出从 P_M 到 P_N 的所有素数，每 10 个数字占 1 行，其间以空格分隔，但行末不得有多余空格。

输入样例:

```
5 27
```

输出样例:

```
11 13 17 19 23 29 31 37 41 43
47 53 59 61 67 71 73 79 83 89
97 101 103
```

```cpp
#include <iostream>
#include <vector>
using namespace std;
bool isprime(int a) {
    for (int i = 2; i * i <= a; i++)
        if(a % i == 0) return false;
    return true;
}
int main() {
    int M, N, num = 2, cnt = 0;
    cin >> M >> N;
    vector<int> v;
    while (cnt < N) {
        if (isprime(num)) {
            cnt++;
            if (cnt >= M) v.push_back(num);
        }
        num++;
    }
    cnt = 0;
    for (int i = 0; i < v.size(); i++) {
        cnt++;
        if (cnt % 10 != 1) printf(" ");
        printf("%d", v[i]);
        if (cnt % 10 == 0) printf("\n");
    }
    return 0;
}
```

1014 福尔摩斯的约会 (20分)

返回

1014 福尔摩斯的约会 (20分)

大侦探福尔摩斯接到一张奇怪的字条：`我们约会吧！ 3485djDkxh4hhGE 2984akDfkkkkggEdsb s&hgsfdk d&Hyscvnm`。大侦探很快就明白了，字条上奇怪的乱码实际上就是约会的时间`星期四 14:04`，因为前面两字符串中第 1 对相同的大写英文字母（大小写有区分）是第 4 个字母`D`，代表星期四；第 2 对相同的字符是`E`，那是第 5 个英文字母，代表一天里的第 14 个钟头（于是一天的 0 点到 23 点由数字 0 到 9、以及大写字母`A`到`N`表示）；后面两字符串第 1 对相同的英文字母`s`出现在第 4 个位置（从 0 开始计数）上，代表第 4 分钟。现给定两对字符串，请帮助福尔摩斯解码得到约会的时间。

输入格式:

输入在 4 行中分别给出 4 个非空、不包含空格、且长度不超过 60 的字符串。

输出格式:

在一行中输出约会的时间，格式为`DAY HH:MM`，其中`DAY`是某星期的 3 字符缩写，即`MON`表示星期一，`TUE`表示星期二，`WED`表示星期三，`THU`表示星期四，`FRI`表示星期五，`SAT`表示星期六，`SUN`表示星期日。题目输入保证每个测试存在唯一解。

输入样例：

```
3485djDkxh4hhGE
2984akDfkkkkggEdsb
s&hgsfdk
d&Hyscvnm
```

输出样例：

```
THU 14:04
```

```
#include <iostream>
#include <cctype>
using namespace std;
int main() {
    string a, b, c, d;
    cin >> a >> b >> c >> d;
    char t[2];
    int pos, i = 0, j = 0;
    while(i < a.length() && i < b.length()) {
        if (a[i] == b[i] && (a[i] >= 'A' && a[i] <= 'G')) {
            t[0] = a[i];
            break;
        }
        i++;
    }
    i = i + 1;
    while (i < a.length() && i < b.length()) {
        if (a[i] == b[i] && ((a[i] >= 'A' && a[i] <= 'N') || isdigit(a[i]))) {
            t[1] = a[i];
            break;
        }
        i++;
    }
    while (j < c.length() && j < d.length()) {
        if (c[j] == d[j] && isalpha(c[j])) {
            pos = j;
            break;
        }
        j++;
    }
    string week[7] = {"MON ", "TUE ", "WED ", "THU ", "FRI ", "SAT ", "SUN "};
    int m = isdigit(t[1]) ? t[1] - '0' : t[1] - 'A' + 10;
    cout << week[t[0]-'A'];
    printf("%02d:%02d", m, pos);
    return 0;
}
```

1015 德才论 (25分)

1015 德才论 (25分)

宋代史学家司马光在《资治通鉴》中有一段著名的“德才论”：“是故才德全尽谓之圣人，才德兼亡谓之愚人，德胜才谓之君子，才胜德谓之小人。凡取人之术，苟不得圣人，君子而与之，与其得小人，不若得愚人。”

现给出一批考生的德才分数，请根据司马光的理论给出录取排名。

输入格式:

输入第一行给出 3 个正整数，分别为：N（$\leq 10^5$），即考生总数；L（≥ 60），为录取最低分数线，即德分和才分均不低于 L 的考生才有资格被考虑录取；H（< 100），为优先录取线——德分和才分均不低于此线的被定义为“才德全尽”，此类考生按德才总分从高到低排序；才分不到但德分到线的一类考生属于“德胜才”，也按总分排序，但排在第一类考生之后；德才分均低于 H，但是德分不低于才分的考生属于“才德兼亡”但尚有“德胜才”者，按总分排序，但排在第二类考生之后；其他达到最低线 L 的考生也按总分排序，但排在第三类考生之后。

随后 N 行，每行给出一位考生的信息，包括：`准考证号 德分 才分`，其中`准考证号`为 8 位整数，德才分为区间 [0, 100] 内的整数。数字间以空格分隔。

输出格式:

输出第一行首先给出达到最低分数线的考生人数 M，随后 M 行，每行按照输入格式输出一位考生的信息，考生按输入中说明的规则从高到低排序。当某类考生中有多人总分相同时，按其德分降序排列；若德分也并列，则按准考证号的升序输出。

输入样例：

```
14 60 80
10000001 64 90
10000002 90 60
10000011 85 80
10000003 85 80
10000004 80 85
10000005 82 77
10000006 83 76
10000007 90 78
10000008 75 79
10000009 59 90
10000010 88 45
10000012 80 100
10000013 90 99
10000014 66 60
```

输出样例：

```
12
10000013 90 99
10000012 80 100
10000003 85 80
10000011 85 80
10000004 80 85
10000007 90 78
10000006 83 76
10000005 82 77
10000002 90 60
10000014 66 60
10000008 75 79
10000001 64 90
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
struct node {
    int num, de, cai;
};
int cmp(struct node a, struct node b) {
    if ((a.de + a.cai) != (b.de + b.cai))
        return (a.de + a.cai) > (b.de + b.cai);
    else if (a.de != b.de)
        return a.de > b.de;
    else
        return a.num < b.num;
}
int main() {
    int n, low, high;
    scanf("%d %d %d", &n, &low, &high);
    vector<node> v[4];
    node temp;
    int total = n;
    for (int i = 0; i < n; i++) {
        scanf("%d %d %d", &temp.num, &temp.de, &temp.cai);
```

```
        if (temp.de < low || temp.cai < low)
            total--;
        else if (temp.de >= high && temp.cai >= high)
            v[0].push_back(temp);
        else if (temp.de >= high && temp.cai < high)
            v[1].push_back(temp);
        else if (temp.de < high && temp.cai < high && temp.de >= temp.cai)
            v[2].push_back(temp);
        else
            v[3].push_back(temp);
    }
    printf("%d\n", total);
    for (int i = 0; i < 4; i++) {
        sort(v[i].begin(), v[i].end(), cmp);
        for (int j = 0; j < v[i].size(); j++)
            printf("%d %d %d\n", v[i][j].num, v[i][j].de, v[i][j].cai);
    }
    return 0;
}
```

1016 部分A+B (15分)

1016 部分A+B (15分)

正整数 A 的“D_A（为 1 位整数）部分”定义为由 A 中所有 D_A 组成的新整数 P_A。例如：给定 $A = 3862767$，$D_A = 6$，则 A 的“6 部分” P_A 是 66，因为 A 中有 2 个 6。

现给定 A、D_A、B、D_B，请编写程序计算 $P_A + P_B$。

输入格式:

输入在一行中依次给出 A、D_A、B、D_B，中间以空格分隔，其中 $0 < A, B < 10^{10}$。

输出格式:

在一行中输出 $P_A + P_B$ 的值。

输入样例 1:

```
3862767 6 13530293 3
```

输出样例 1:

```
399
```

输入样例 2:

```
3862767 1 13530293 8
```

输出样例 2:

```
0
```

```cpp
#include <iostream>

using namespace std;

long long f(string a,char b)
{
    int cnt = 0;
    for(auto c : a)
    {
        if(c == b) cnt ++;
    }

    long long res = 0;
    string s;
    while(cnt --)
    {
        s += b;
    }
    res = atoi(s.c_str());
    return res;
}
```

```
int main()
{
    string a,b;
    char Da,Db;
    cin >> a >> Da >> b >> Db;
    cout << f(a,Da) + f(b,Db);

}
```

1017 A除以B (20分)

‹ 返回

1017 A除以B (20分)

本题要求计算 A/B，其中 A 是不超过 1000 位的正整数，B 是 1 位正整数。你需要输出商数 Q 和余数 R，使得 $A = B \times Q + R$ 成立。

输入格式:

输入在一行中依次给出 A 和 B，中间以 1 空格分隔。

输出格式:

在一行中依次输出 Q 和 R，中间以 1 空格分隔。

输入样例:

```
123456789050987654321 7
```

输出样例:

```
17636684150141093474 3
```

```
#include <iostream>
using namespace std;
int main() {
    string s;
    int a, t = 0, temp = 0;
    cin >> s >> a;
    int len = s.length();
    t = (s[0] - '0') / a;
    if ((t != 0 && len > 1) || len == 1)
        cout << t;
    temp = (s[0] - '0') % a;
    for (int i = 1; i < len; i++) {
        t = (temp * 10 + s[i] - '0') / a;
        cout << t;
        temp = (temp * 10 + s[i] - '0') % a;
    }
    cout << " " << temp;
    return 0;
```

```
}
```

1018 锤子剪刀布 (20分)

大家应该都会玩“锤子剪刀布”的游戏：两人同时给出手势，胜负规则如图所示：

现给出两人的交锋记录，请统计双方的胜、平、负次数，并且给出双方分别出什么手势的胜算最大。

输入格式:

输入第 1 行给出正整数 N （$\leq 10^5$），即双方交锋的次数。随后 N 行，每行给出一次交锋的信息，即甲、乙双方同时给出的的手势。`C` 代表“锤子”、`J` 代表“剪刀”、`B` 代表“布”，第 1 个字母代表甲方，第 2 个代表乙方，中间有 1 个空格。

输出格式:

输出第 1、2 行分别给出甲、乙的胜、平、负次数，数字间以 1 个空格分隔。第 3 行给出两个字母，分别代表甲、乙获胜次数最多的手势，中间有 1 个空格。如果解不唯一，则输出按字母序最小的解。

输入样例:

```
10
C J
J B
C B
B B
B C
C C
C B
J B
B C
J J
```

输出样例:

```
5 3 2
2 3 5
B B
```

```cpp
#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    int jiawin = 0, yiwin = 0;
    int jia[3] = {0}, yi[3] = {0};
    for (int i = 0; i < n; i++) {
        char s, t;
        cin >> s >> t;
        if (s == 'B' && t == 'C') {
            jiawin++;
            jia[0]++;
        } else if (s == 'B' && t == 'J') {
            yiwin++;
            yi[2]++;
        } else if (s == 'C' && t == 'B') {
            yiwin++;
            yi[0]++;
        } else if (s == 'C' && t == 'J') {
            jiawin++;
            jia[1]++;
        } else if (s == 'J' && t == 'B') {
            jiawin++;
            jia[2]++;
        } else if (s == 'J' && t == 'C') {
            yiwin++;
            yi[1]++;
        }
    }
    cout << jiawin << " " << n - jiawin - yiwin << " " << yiwin << endl << yiwin
<< " " << n - jiawin - yiwin << " " << jiawin << endl;
    int maxjia = jia[0] >= jia[1] ? 0 : 1;
    maxjia = jia[maxjia] >= jia[2] ? maxjia : 2;
```

```
    int maxyi = yi[0] >= yi[1] ? 0 : 1;
    maxyi = yi[maxyi] >= yi[2] ? maxyi : 2;
    char str[4] = {"BCJ"};
    cout << str[maxjia] << " " << str[maxyi];
    return 0;
}
```

1019 数字黑洞 (20分)

1019 数字黑洞 (20分)

给定任一个各位数字不完全相同的 4 位正整数，如果我们先把 4 个数字按非递增排序，再按非递减排序，然后用第 1 个数字减第 2 个数字，将得到一个新的数字。一直重复这样做，我们很快会停在有“数字黑洞”之称的 `6174`，这个神奇的数字也叫 Kaprekar 常数。

例如，我们从 `6767` 开始，将得到

```
7766 - 6677 = 1089
9810 - 0189 = 9621
9621 - 1269 = 8352
8532 - 2358 = 6174
7641 - 1467 = 6174
... ...
```

现给定任意 4 位正整数，请编写程序演示到达黑洞的过程。

输入格式:

输入给出一个 $(0, 10^4)$ 区间内的正整数 N。

输出格式:

如果 N 的 4 位数字全相等，则在一行内输出 `N - N = 0000`；否则将计算的每一步在一行内输出，直到 `6174` 作为差出现，输出格式见样例。注意每个数字按 `4` 位数格式输出。

输入样例 1:

```
6767
```

输出样例 1:

```
7766 - 6677 = 1089
9810 - 0189 = 9621
9621 - 1269 = 8352
8532 - 2358 = 6174
```

输入样例 2:

```
2222
```

输出样例 2:

```
2222 - 2222 = 0000
```

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
bool cmp(char a, char b) {return a > b;}
int main() {
    string s;
    cin >> s;
    s.insert(0, 4 - s.length(), '0');
    do {
        string a = s, b = s;
        sort(a.begin(), a.end(), cmp);
        sort(b.begin(), b.end());
        int result = stoi(a) - stoi(b);
        s = to_string(result);
        s.insert(0, 4 - s.length(), '0');
        cout << a << " - " << b << " = " << s << endl;
    } while (s != "6174" && s != "0000");
    return 0;
}
```

1020 月饼 (25分)

1020 月饼 (25分)

月饼是中国人在中秋佳节时吃的一种传统食品，不同地区有许多不同风味的月饼。现给定所有种类月饼的库存量、总售价、以及市场的最大需求量，请你计算可以获得的最大收益是多少。

注意：销售时允许取出一部分库存。样例给出的情形是这样的：假如我们有 3 种月饼，其库存量分别为 18、15、10 万吨，总售价分别为 75、72、45 亿元。如果市场的最大需求量只有 20 万吨，那么我们最大收益策略应该是卖出全部 15 万吨第 2 种月饼、以及 5 万吨第 3 种月饼，获得 72 + 45/2 = 94.5（亿元）。

输入格式:

每个输入包含一个测试用例。每个测试用例先给出一个不超过 1000 的正整数 N 表示月饼的种类数、以及不超过 500（以万吨为单位）的正整数 D 表示市场最大需求量。随后一行给出 N 个正数表示每种月饼的库存量（以万吨为单位）；最后一行给出 N 个正数表示每种月饼的总售价（以亿元为单位）。数字间以空格分隔。

输出格式:

对每组测试用例，在一行中输出最大收益，以亿元为单位并精确到小数点后 2 位。

输入样例:

```
3 20
18 15 10
75 72 45
```

输出样例:

```
94.50
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
struct mooncake{
    float mount, price, unit;
};
int cmp(mooncake a, mooncake b) {
    return a.unit > b.unit;
}
int main() {
    int n, need;
    cin >> n >> need;
    vector<mooncake> a(n);
    for (int i = 0; i < n; i++) scanf("%f", &a[i].mount);
    for (int i = 0; i < n; i++) scanf("%f", &a[i].price);
    for (int i = 0; i < n; i++) a[i].unit = a[i].price / a[i].mount;
    sort(a.begin(), a.end(), cmp);
    float result = 0.0;
    for (int i = 0; i < n; i++) {
        if (a[i].mount <= need) {
            result = result + a[i].price;
```

```
        } else {
            result = result + a[i].unit * need;
            break;
        }
        need = need - a[i].mount;
    }
    printf("%.2f",result);
    return 0;
}
```