

1041 考试座位号 (15分)



1041 考试座位号 (15分)

每个 PAT 考生在参加考試時都會被分配兩個座位號，一個是試機座位，一個是考試座位。正常情況下，考生在入場時先得到試機座位號碼，入座進入試機狀態後，系統會顯示該考生的考試座位號碼，考試時考生需要換到考試座位就座。但有些考生遲到了，試機已經結束，他們只能拿着領到的試機座位號碼求助於你，從後台查出他們的考試座位號碼。

輸入格式：

輸入第一行給出一個正整數 N (≤ 1000)，隨後 N 行，每行給出一個考生的信息：`准考证号 试机座位号 考试座位号`。其中 `准考证号` 由 16 位數字組成，座位從 1 到 N 編號。輸入保證每個人的准考证号都不同，並且任何時候都不會把兩個人分配到同一個座位上。

考生信息之後，給出一個正整數 M ($\leq N$)，隨後一行中給出 M 個待查詢的試機座位號碼，以空格分隔。

輸出格式：

對應每個需要查詢的試機座位號碼，在一行中輸出對應考生的准考证号和考試座位號碼，中間用 1 個空格分隔。

輸入样例：

```
4
3310120150912233 2 4
3310120150912119 4 1
3310120150912126 1 3
3310120150912002 3 2
2
3 4
```

輸出样例：

```
3310120150912002 2
3310120150912119 1
```

```
// #include <iostream>

// using namespace std;

// struct s
// {
//     int id,a,b;
// }s[1000];
// int main()
// {
//     int n;
//     cin >> n;
//     for(int i = 0 ;i < n;i ++ ) cin >> s[i].id >> s[i].a >> s[i].b;
```

```

//      int t;
//      cin >> t;
//      while(t --)
//      {
//          int cur_a;
//          cin >> cur_a;
//          for(int i = 0;i < n;i ++)
//          {
//              if(s[i].a == cur_a)
//              {
//                  cout << s[i].id <<' ' << s[i].b << endl;
//                  break;
//              }
//          }
//      }
//  }
// }
#include <iostream>
using namespace std;
int main() {
    string stu[1005][2], s1, s2;;
    int n, m, t;
    cin >> n;
    for(int i = 0; i < n; i++) {
        cin >> s1 >> t >> s2;
        stu[t][0] = s1;
        stu[t][1] = s2;
    }
    cin >> m;
    for(int i = 0; i < m; i++) {
        cin >> t;
        cout << stu[t][0] << " " << stu[t][1] << endl;
    }
    return 0;
}

```

1042 字符统计 (20分)

请编写程序，找出一段给定文字中出现最频繁的那个英文字母。

输入格式：

输入在一行中给出一个长度不超过 1000 的字符串。字符串由 ASCII 码表中任意可见字符及空格组成，至少包含 1 个英文字母，以回车结束（回车不算在内）。

输出格式：

在一行中输出出现频率最高的那个英文字母及其出现次数，其间以空格分隔。如果有并列，则输出按字母序最小的那个字母。统计时不区分大小写，输出小写字母。

输入样例：

```
This is a simple TEST.  There ARE numbers and other symbols 1&2&3.....
```

输出样例：

```
e 7
```

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string s1;
    int max = 0;
    int h[26] = {0};
    char res;
    getline(cin,s1);
    for(auto &c:s1)
    {
        c = tolower(c);
        if(c >= 'a' && c <='z')
            h[c - 'a'] ++;
    }
    for(int i = 0;i < 26;i ++ )
    {
        if(h[i] > max)
        {
            max = h[i];
            res = i + 'a';
        }
    }
    cout << res << ' ' << max;

}
```

1043 输出PATest (20分)

[← 返回](#)

1043 输出PATest (20分)

给定一个长度不超过 10^4 的、仅由英文字母构成的字符串。请将字符重新调整顺序，按 `PATestPATest...` 这样的顺序输出，并忽略其它字符。当然，六种字符的个数不一定是一样多的，若某种字符已经输出完，则余下的字符仍按 PATest 的顺序打印，直到所有字符都被输出。

输入格式：

输入在一行中给出一个长度不超过 10^4 的、仅由英文字母构成的非空字符串。

输出格式：

在一行中按题目要求输出排序后的字符串。题目保证输出非空。

输入样例：

```
redlesPayBestPATtopTeePHPereaititAPPT
```

输出样例：

```
PATestPATestPTetPTePePee
```

```
#include <iostream>

using namespace std;

int main()
{
    string s1;
    int h[6] = {0};
    getline(cin,s1);
    for(auto c:s1)
    {
        if(c == 'P') h[0]++;
        if(c == 'A') h[1]++;
        if(c == 'T') h[2]++;
        if(c == 'e') h[3]++;
        if(c == 's') h[4]++;
        if(c == 't') h[5]++;
    }
    while (h[0] > 0 || h[1] > 0 || h[2] > 0 || h[3] > 0 || h[4] > 0 || h[5] > 0)
    {
        if(h[0]-- > 0) cout <<'P';
        if(h[1]-- > 0) cout <<'A';
        if(h[2]-- > 0) cout <<'T';
        if(h[3]-- > 0) cout <<'e';
        if(h[4]-- > 0) cout <<'s';
        if(h[5]-- > 0) cout <<'t';
```

```
}  
}
```

1044 火星数字 (20分)

[← 返回](#)

1044 火星数字 (20分)

火星人是 以 13 进制计数的：

- 地球人的 0 被火星人称为 tret。
- 地球人数字 1 到 12 的火星文分别为：jan, feb, mar, apr, may, jun, jly, aug, sep, oct, nov, dec。
- 火星将进位以后的 12 个高位数字分别称为：tam, hel, maa, huh, tou, kes, hei, elo, syy, lok, mer, jou。

例如地球人的数字 29 翻译成火星文就是 hel mar；而火星文 elo nov 对应地球数字 115。为了方便交流，请你编写程序实现地球和火星数字之间的互译。

输入格式：

输入第一行给出一个正整数 N (< 100)，随后 N 行，每行给出一个 $[0, 169)$ 区间内的数字——或者是地球文，或者是火星文。

输出格式：

对应输入的每一行，在一行中输出翻译后的另一种语言的数字。

输入样例：

```
4  
29  
5  
elo nov  
tam
```

输出样例：

```
hel mar  
may  
115  
13
```

```
#include <iostream>  
#include <string>  
using namespace std;  
string a[13] = {"tret", "jan", "feb", "mar", "apr", "may", "jun", "jly", "aug",  
"sep", "oct", "nov", "dec"};  
string b[13] = {"####", "tam", "hel", "maa", "huh", "tou", "kes", "hei", "elo",  
"syy", "lok", "mer", "jou"};  
string s;  
int len;  
void func1(int t) {  
    if (t / 13) cout << b[t / 13];
```

```

        if ((t / 13) && (t % 13)) cout << " ";
        if (t % 13 || t == 0) cout << a[t % 13];
    }
    void func2() {
        int t1 = 0, t2 = 0;
        string s1 = s.substr(0, 3), s2;
        if (len > 4) s2 = s.substr(4, 3);
        for (int j = 1; j <= 12; j++) {
            if (s1 == a[j] || s2 == a[j]) t2 = j;
            if (s1 == b[j]) t1 = j;
        }
        cout << t1 * 13 + t2;
    }
    int main() {
        int n;
        cin >> n;
        getchar();
        for (int i = 0; i < n; i++) {
            getline(cin, s);
            len = s.length();
            if (s[0] >= '0' && s[0] <= '9')
                func1(stoi(s));
            else
                func2();
            cout << endl;
        }
        return 0;
    }
}

```

1045 快速排序 (25分)

著名的快速排序算法里有一个经典的划分过程：我们通常采用某种方法取一个元素作为主元，通过交换，把比主元小的元素放到它的左边，比主元大的元素放到它的右边。给定划分后的 N 个互不相同的正整数的排列，请问有多少个元素可能是划分前选取的主元？

例如给定 $N = 5$ ，排列是 1、3、2、4、5。则：

- 1 的左边没有元素，右边的元素都比它大，所以它可能是主元；
- 尽管 3 的左边元素都比它小，但其右边的 2 比它小，所以它不能是主元；
- 尽管 2 的右边元素都比它大，但其左边的 3 比它大，所以它不能是主元；
- 类似原因，4 和 5 都可能是主元。

因此，有 3 个元素可能是主元。

输入格式：

输入在第 1 行中给出一个正整数 N ($\leq 10^5$)；第 2 行是空格分隔的 N 个不同的正整数，每个数不超过 10^9 。

输出格式：

在第 1 行中输出有可能是主元的元素个数；在第 2 行中按递增顺序输出这些元素，其间以 1 个空格分隔，行首尾不得有多余空格。

输入样例：

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

输出样例：

```
3
1 4 5
```

```
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

const int N = 1e5 + 10;
int v[N];

int main() {
    int n, max = 0, cnt = 0;
    cin >> n;
    vector<int> a(n), b(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
        b[i] = a[i];
    }
    sort(a.begin(), a.end());
    for (int i = 0; i < n; i++) {
```

```

        if(a[i] == b[i] && b[i] > max)
            v[cnt++] = b[i];
        if (b[i] > max)
            max = b[i];
    }
    cout << cnt << endl;
    for(int i = 0; i < cnt; i++) {
        if (i != 0) cout << ' ';
        cout << v[i];
    }
    cout << endl;
    return 0;
}

```

1046 划拳 (15分)



1046 划拳 (15分)

划拳是古老中国酒文化的一个有趣的组成部分。酒桌上两人划拳的方法为：每人口中喊出一个数字，同时用手比划出一个数字。如果谁比划出的数字正好等于两人喊出的数字之和，谁就赢了，输家罚一杯酒。两人同赢或两人同输则继续下一轮，直到唯一的赢家出现。

下面给出甲、乙两人的划拳记录，请你统计他们最后分别喝了多少杯酒。

输入格式：

输入第一行先给出一个正整数 N (≤ 100)，随后 N 行，每行给出一轮划拳的记录，格式为：

甲喊 甲划 乙喊 乙划

其中 **喊** 是喊出的数字，**划** 是划出的数字，均为不超过 100 的正整数（两只手一起划）。

输出格式：

在一行中先后输出甲、乙两人喝酒的杯数，其间以一个空格分隔。

输入样例：

```

5
8 10 9 12
5 10 5 10
3 8 5 12
12 18 1 13
4 16 12 15

```

输出样例：

```

1 2

```

```

#include <iostream>

using namespace std;

```



```

int res[2];
int main()
{
    int n;
    cin >> n;
    while(n --)
    {
        int a1,a2,b1,b2;
        cin >> a1 >> a2 >> b1 >> b2;
        if(a2 != b2)
        {
            if(a2 == (a1 + b1)) res[1] ++;
            if(b2 == (a1 + b1)) res[0] ++;
        }
    }
    cout << res[0] << ' ' << res[1];
}

```

1047 编程团体赛 (20分)

[返回](#)

1047 编程团体赛 (20分)

编程团体赛的规则为：每个参赛队由若干队员组成；所有队员独立比赛；参赛队的成绩为所有队员的成绩和；成绩最高的队获胜。

现给定所有队员的比赛成绩，请你编写程序找出冠军队。

输入格式：

输入第一行给出一个正整数 N ($\leq 10^4$)，即所有参赛队员总数。随后 N 行，每行给出一位队员的成绩，格式为：`队伍编号-队员编号 成绩`，其中 `队伍编号` 为 1 到 1000 的正整数，`队员编号` 为 1 到 10 的正整数，`成绩` 为 0 到 100 的整数。

输出格式：

在一行中输出冠军队的编号和总成绩，其间以一个空格分隔。注意：题目保证冠军队是唯一的。

输入样例：

```

6
3-10 99
11-5 87
102-1 0
102-3 100
11-9 89
3-2 61

```

输出样例：

```

11 176

```

```
#include <iostream>
```

```

#include <string>
using namespace std;

int res[10010];

int main()
{
    int n;
    int max = -1;
    int num;
    cin >> n;
    while(n --)
    {
        int t,id;
        int score;
        scanf("%d-%d %d", &t, &id, &score);
        res[t] += score;
    }
    for(int i = 0;i < 10010;i ++)
    {
        if(res[i] > max) max = res[i],num = i;
    }
    cout << num << ' ' << max;
}

```

1048 数字加密 (20分)

[返回](#)

1048 数字加密 (20分)

本题要求实现一种数字加密方法。首先固定一个加密用正整数 A，对任一正整数 B，将其每 1 位数字与 A 的对应位置上的数字进行以下运算：对奇数位，对应位数字相加后对 13 取余——这里用 J 代表 10、Q 代表 11、K 代表 12；对偶数位，用 B 的数字减去 A 的数字，若结果为负数，则再加 10。这里令个位为第 1 位。

输入格式：

输入在一行中依次给出 A 和 B，均为不超过 100 位的正整数，其间以空格分隔。

输出格式：

在一行中输出加密后的结果。

输入样例：

```
1234567 368782971
```

输出样例：

```
3695Q8118
```

```
#include <iostream>
```

```

#include <string>

using namespace std;

int main()
{
    string s1,s2;
    cin >> s1 >> s2;

    for(int i = s1.size() - 1,j = 1;i >= 0;i --,j ++){
        if(j % 2 != 0)
            s1[i] = ((s2[i] - '0') - (s1[i] - '0')) > 0 ? (s2[i] - '0') - (s1[i] - '0') + '0': 10 + (s2[i] - '0') - (s1[i] - '0') + '0';
        else
        {
            char res;
            int s = (s1[i] - '0') + (s2[i] - '0');
            s = s % 13;
            res = s + '0';
            if(s == 10) res = 'J';
            else if(s == 11) res = 'Q';
            else if(s == 12) res = 'K';
            s1[i] = res;
        }
        for(int i= 0;i < s1.size();i ++)    cout << s1[i];
    }
}

```

1049 数列的片段和 (20分)

给定一个正数数列，我们可以从中截取任意的连续的几个数，称为片段。例如，给定数列 { 0.1, 0.2, 0.3, 0.4 }，我们有 (0.1) (0.1, 0.2) (0.1, 0.2, 0.3) (0.1, 0.2, 0.3, 0.4) (0.2) (0.2, 0.3) (0.2, 0.3, 0.4) (0.3) (0.3, 0.4) (0.4) 这 10 个片段。

给定正整数数列，求出全部片段包含的所有的数之和。如本例中 10 个片段总和是 $0.1 + 0.3 + 0.6 + 1.0 + 0.2 + 0.5 + 0.9 + 0.3 + 0.7 + 0.4 = 5.0$ 。

输入格式：

输入第一行给出一个不超过 10^5 的正整数 N ，表示数列中数的个数，第二行给出 N 个不超过 1.0 的正数，是数列中的数，其间以空格分隔。

输出格式：

在一行中输出该序列所有片段包含的数之和，精确到小数点后 2 位。

输入样例：

```
4
0.1 0.2 0.3 0.4
```

输出样例：

```
5.00
```

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    double sum = 0.0, temp;
    for (int i = 1; i <= n; i++) {
        cin >> temp;
        sum = sum + temp * i * (n - i + 1);
    }
    printf("%.2f", sum);
    return 0;
}
```

1050 螺旋矩阵 (25分)

本题要求将给定的 N 个正整数按非递增的顺序，填入“螺旋矩阵”。所谓“螺旋矩阵”，是指从左上角第 1 个格子开始，按顺时针螺旋方向填充。要求矩阵的规模为 m 行 n 列，满足条件： $m \times n$ 等于 N ； $m \geq n$ ；且 $m - n$ 取所有可能值中的最小值。

输入格式：

输入在第 1 行中给出一个正整数 N ，第 2 行给出 N 个待填充的正整数。所有数字不超过 10^4 ，相邻数字以空格分隔。

输出格式：

输出螺旋矩阵。每行 n 个数字，共 m 行。相邻数字以 1 个空格分隔，行末不得有多余空格。

输入样例：

```
12
37 76 20 98 76 42 53 95 60 81 58 93
```

输出样例：

```
98 95 93
42 37 81
53 20 76
58 60 76
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>

using namespace std;

const int N = 110;
int st[N][N];
int res[N][N];

int cmp(int a, int b) {return a > b;}

int main()
{
    int N, m, n, t = 0;
    cin >> N;
    for (m = sqrt((double)N); m >= 1; m--)
    {
        if (N % m == 0)
        {
            n = N / m;
            break;
        }
    }
```

```

}
vector<int> s(N);
for(int i = 0; i < N; i++) cin >> s[i];
sort(s.begin(), s.end(), cmp);

int x = 0, y = 0, d = 0;

int dx[] = {0, 1, 0, -1}, dy[] = {1, 0, -1, 0};

for(int i = 1; i <= n * m; i++)
{
    int nx = x + dx[d], ny = y + dy[d];
    if(nx < 0 || nx >= n || ny < 0 || ny >= m || st[nx][ny])
    {
        d = (d + 1) % 4;
        nx = x + dx[d];
        ny = y + dy[d];
    }
    res[x][y] = s[i - 1];
    st[x][y] = true;
    x = nx;
    y = ny;
}

for(int i = 0; i < n; i++)
{
    for(int j = 0; j < m; j++) cout << res[i][j] << ' ';
    cout << endl;
}
}

```

1051 复数乘法 (15分)

复数可以写成 $(A + Bi)$ 的常规形式，其中 A 是实部， B 是虚部， i 是虚数单位，满足 $i^2 = -1$ ；也可以写成极坐标下的指数形式 $(R \times e^{(Pi)})$ ，其中 R 是复数模， P 是辐角， i 是虚数单位，其等价于三角形式 $R(\cos(P) + i \sin(P))$ 。

现给定两个复数的 R 和 P ，要求输出两数乘积的常规形式。

输入格式：

输入在一行中依次给出两个复数的 R_1, P_1, R_2, P_2 ，数字间以空格分隔。

输出格式：

在一行中按照 `A+Bi` 的格式输出两数乘积的常规形式，实部和虚部均保留 2 位小数。注意：如果 B 是负数，则应该写成 `A-|B|i` 的形式。

输入样例：

```
2.3 3.5 5.2 0.4
```

输出样例：

```
-8.68-8.23i
```

```
#include <iostream>
#include <cmath>

using namespace std;

int main() {
    double r1, p1, r2, p2, A, B;
    cin >> r1 >> p1 >> r2 >> p2;
    A = r1 * r2 * cos(p1) * cos(p2) - r1 * r2 * sin(p1) * sin(p2);
    B = r1 * r2 * cos(p1) * sin(p2) + r1 * r2 * sin(p1) * cos(p2);
    if (A + 0.005 >= 0 && A < 0)
        printf("0.00");
    else
        printf("%.2f", A);
    if (B >= 0)
        printf("+%.2fi", B);
    else if (B + 0.005 >= 0 && B < 0)
        printf("+0.00i");
    else
        printf("%.2fi", B);
    return 0;
}
```

1052 卖个萌 (20分)

萌萌哒表情符号通常由“手”、“眼”、“口”三个主要部分组成。简单起见，我们假设一个表情符号是按下列格式输出的：

```
[左手]([左眼][口][右眼])[右手]
```

现给出可选用的符号集合，请你按用户的要求输出表情。

输入格式：



输入首先在前三行顺序对应给出手、眼、口的可选符号集。每个符号括在一对方括号 `[]` 内。题目保证每个集合都至少有一个符号，并不超过 10 个符号；每个符号包含 1 到 4 个非空字符。

之后一行给出一个正整数 K ，为用户请求的个数。随后 K 行，每行给出一个用户的符号选择，顺序为左手、左眼、口、右眼、右手——这里只给出符号在相应集合中的序号（从 1 开始），数字间以空格分隔。


输出格式：

对每个用户请求，在一行中输出生成的表情。若用户选择的序号不存在，则输出 `Are you kidding me?` @\@。

1053 住房空置率 (20分)

在不打扰居民的前提下，统计住房空置率的一种方法是根据每户用电量的连续变化规律进行判断。判断方法如下：

- 在观察期内，若存在超过一半的日子用电量低于某给定的阈值 e ，则该住房为“可能空置”；
- 若观察期超过某给定阈值 D 天，且满足上一个条件，则该住房为“空置”。

现给定某居民区的住户用电量数据，请你统计“可能空置”比率和“空置”比率，即以上两种状态的住房占居民区住房总套数的百分比。

输入格式：

输入第一行给出正整数 N (≤ 1000)，为居民区住房总套数；正实数 e ，即低电量阈值；正整数 D ，即观察期阈值。随后 N 行，每行按以下格式给出一套住房的用电量数据：

$K \ E_1 \ E_2 \ \dots \ E_K$

其中 K 为观察的天数， E_i 为第 i 天的用电量。

输出格式：

在一行中输出“可能空置”的比率和“空置”比率的百分比值，其间以一个空格分隔，保留小数点后 1 位。

输入样例：

```
5 0.5 10
6 0.3 0.4 0.5 0.2 0.8 0.6
10 0.0 0.1 0.2 0.3 0.0 0.8 0.6 0.7 0.0 0.5
5 0.4 0.3 0.5 0.1 0.7
11 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
11 2 2 2 1 1 0.1 1 0.1 0.1 0.1 0.1
```

输出样例：

```
40.0% 20.0%
```

(样例解释：第2、3户为“可能空置”，第4户为“空置”，其他户不是空置。)

```
#include <iostream>
using namespace std;
int main() {
    int n, d, k, maybe = 0, must = 0;
    double e, temp;
    cin >> n >> e >> d;
    for (int i = 0; i < n; i++) {
        cin >> k;
        int sum = 0;
        for (int j = 0; j < k; j++) {
            cin >> temp;
            if (temp < e) sum++;
        }
        if (sum > (k / 2)) {
            k > d ? must++ : maybe++;
        }
    }
    double mayberesult = (double)maybe / n * 100;
    double mustresult = (double)must / n * 100;
    printf("%.1f%% %.1f%%", mayberesult, mustresult);
    return 0;
}
```

1054 求平均值 (20分)

本题的基本要求非常简单：给定 N 个实数，计算它们的平均值。但复杂的是有些输入数据可能是非法的。一个“合法”的输入是 $[-1000, 1000]$ 区间内的实数，并且最多精确到小数点后 2 位。当你计算平均值的时候，不能把那些非法的数据算在内。

输入格式：

输入第一行给出正整数 N (≤ 100)。随后一行给出 N 个实数，数字间以一个空格分隔。

输出格式：

对每个非法输入，在一行中输出 `ERROR: X is not a legal number`，其中 `X` 是输入。最后在一行中输出结果：`The average of K numbers is Y`，其中 `K` 是合法输入的个数，`Y` 是它们的平均值，精确到小数点后 2 位。如果平均值无法计算，则用 `Undefined` 替换 `Y`。如果 `K` 为 1，则输出 `The average of 1 number is Y`。

输入样例 1：

```
7
5 -3.2 aaa 9999 2.3.4 7.123 2.35
```

输出样例 1：

```
ERROR: aaa is not a legal number
ERROR: 9999 is not a legal number
ERROR: 2.3.4 is not a legal number
ERROR: 7.123 is not a legal number
The average of 3 numbers is 1.38
```

输入样例 2：

```
2
aaa -9999
```

```
#include <iostream>
#include <cstdio>
#include <string.h>
using namespace std;
int main() {
    int n, cnt = 0;
    char a[50], b[50];
    double temp, sum = 0.0;
    cin >> n;
    for(int i = 0; i < n; i++) {
        scanf("%s", a);
        sscanf(a, "%lf", &temp);
        sprintf(b, "%.2f", temp);
        int flag = 0;
        for(int j = 0; j < strlen(a); j++)
            if(a[j] != b[j]) flag = 1;
        if(flag || temp < -1000 || temp > 1000) {
```

```

        printf("ERROR: %s is not a legal number\n", a);
        continue;
    } else {
        sum += temp;
        cnt++;
    }
}
if(cnt == 1)
    printf("The average of 1 number is %.2f", sum);
else if(cnt > 1)
    printf("The average of %d numbers is %.2f", cnt, sum / cnt);
else
    printf("The average of 0 numbers is Undefined");
return 0;
}

```

1055 集体照 (25分)

[返回](#)

1055 集体照 (25分)

拍集体照时队形很重要，这里对给定的 N 个人 K 排的队形设计排队规则如下：

- 每排人数为 N/K （向下取整），多出来的人全部站在最后一排；
- 后排所有人的个子都不比前排任何人矮；
- 每排中最高者站中间（中间位置为 $m/2 + 1$ ，其中 m 为该排人数，除法向下取整）；
- 每排其他人以中间人为轴，按身高非增序，先右后左交替入队站在中间人的两侧（例如5人身高为190、188、186、175、170，则队形为175、188、190、186、170。这里假设你面对拍照者，所以你的左边是中间人的右边）；
- 若多人身高相同，则按名字的字典序升序排列。这里保证无重名。

现给定一组拍照人，请编写程序输出他们的队形。

输入格式：

每个输入包含 1 个测试用例。每个测试用例第 1 行给出两个正整数 N ($\leq 10^4$ ，总人数) 和 K (≤ 10 ，总排数)。随后 N 行，每行给出一个人的名字（不包含空格、长度不超过 8 个英文字母）和身高 ($[30, 300]$ 区间内的整数)。

输出格式：

输出拍照的队形。即 K 排人名，其间以空格分隔，行末不得有多余空格。注意：假设你面对拍照者，后排的人输出在上方，前排输出在下方。

输入样例:

```
10 3
Tom 188
Mike 170
Eva 168
Tim 160
Joe 190
Ann 168
Bob 175
Nick 186
Amy 160
John 159
```

输出样例:

```
Bob Tom Joe Nick
Ann Mike Eva
Tim Amy John
```

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
struct node {
    string name;
    int height;
};
int cmp(struct node a, struct node b) {
    return a.height != b.height ? a.height > b.height : a.name < b.name;
}
int main() {
    int n, k, m;
    cin >> n >> k;
    vector<node> stu(n);
    for(int i = 0; i < n; i++) {
        cin >> stu[i].name >> stu[i].height;
    }
    sort(stu.begin(), stu.end(), cmp);
    int t = 0, row = k;
    while(row) {
        if(row == k)
            m = n - n / k * (k - 1);
        else
            m = n / k;
        vector<string> ans(m);
        ans[m / 2] = stu[t].name;
        // 左边一列
        int j = m / 2 - 1;
        for(int i = t + 1; i < t + m; i = i + 2)
            ans[j--] = stu[i].name;
        // 右边一列
        j = m / 2 + 1;
        for(int i = t + 2; i < t + m; i = i + 2)
            ans[j++] = stu[i].name;
```

```

        // 输出当前排
        cout << ans[0];
        for(int i = 1; i < m; i++)
            cout << " " << ans[i];
        cout << endl;
        t = t + m;
        row--;
    }
    return 0;
}

```

1056 组合数的和 (15分)

[返回](#)

1056 组合数的和 (15分)

给定 N 个非 0 的个位数字，用其中任意 2 个数字都可以组合成 1 个 2 位的数字。要求所有可能组合出来的 2 位数字的和。例如给定 2、5、8，则可以组合出：25、28、52、58、82、85，它们的和为 330。

输入格式：

输入在一行中先给出 N ($1 < N < 10$)，随后给出 N 个不同的非 0 个位数字。数字间以空格分隔。

输出格式：

输出所有可能组合出来的 2 位数字的和。

输入样例：

3 2 8 5

输出样例：

330

```

#include <stdio>
int main() {
    int N, sum = 0, temp;
    scanf("%d", &N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &temp);
        sum += temp * 10 * (N - 1) + temp * (N - 1);
    }
    printf("%d", sum);
    return 0;
}

```

1057 数零壹 (20分)

给定一串长度不超过 10^5 的字符串，本题要求你将其中所有英文字母的序号（字母 a-z 对应序号 1-26，不分大小写）相加，得到整数 N，然后再分析一下 N 的二进制表示中有多少 0、多少 1。例如给定字符串 `PAT (Basic)`，其字母序号之和为：16+1+20+2+1+19+9+3=71，而 71 的二进制是 1000111，即有 3 个 0、4 个 1。

输入格式：

输入在一行中给出长度不超过 10^5 、以回车结束的字符串。

输出格式：

在一行中先后输出 0 的个数和 1 的个数，其间以空格分隔。

输入样例：

```
PAT (Basic)
```

输出样例：

```
3 4
```

```
#include <iostream>

using namespace std;

int main()
{
    string s;
    getline(cin,s);
    int sum;
    for(auto &c:s)
    {
        if(c >= 'A' && c <= 'Z') c += 32;
        if(c >= 'a' && c <= 'z')sum += c - 'a' + 1;
    }
    int res_1 = 0;
    int res_0 = 0;
    while(sum != 0) {
        if(sum % 2 == 0) {
            res_0++;
        } else {
            res_1++;
        }
        sum = sum / 2;
    }
    printf("%d %d", res_0, res_1);
    return 0;
}
```

1058 选择题 (20分)

[< 返回](#)

1058 选择题 (20分)

批改多选题是比较麻烦的事情，本题就请你写个程序帮助老师批改多选题，并且指出哪道题错的人最多。

输入格式：

输入在第一行给出两个正整数 N (≤ 1000) 和 M (≤ 10)，分别是学生人数和多选题的个数。随后 M 行，每行顺次给出一道题的满分值（不超过 5 的正整数）、选项个数（不少于 2 且不超过 5 的正整数）、正确选项个数（不超过选项个数的正整数）、所有正确选项。注意每题的选项从小写英文字母 a 开始顺次排列。各项间以 1 个空格分隔。最后 N 行，每行给出一个学生的答题情况，其每题答案格式为 (选中的选项个数 选项1)，按题目顺序给出。注意：题目保证学生的答题情况是合法的，即不存在选中的选项数超过实际选项数的情况。

输出格式：

按照输入的顺序给出每个学生的得分，每个分数占一行。注意判题时只有选择全部正确才能得到该题的分数。最后一行输出错得最多的题目的错误次数和编号（题目按照输入的顺序从 1 开始编号）。如果有并列，则按编号递增顺序输出。数字间用空格分隔，行首尾不得有多余空格。如果所有题目都没有人错，则在最后一行输出 `Too simple`。

输入样例：

```
3 4
3 4 2 a c
2 5 1 b
5 3 2 b c
1 5 4 a b d e
(2 a c) (2 b d) (2 a c) (3 a b e)
(2 a c) (1 b) (2 a b) (4 a b d e)
(2 b d) (1 e) (2 b c) (4 a b c d)
```

输出样例：

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

1059 C语言竞赛 (20分)

C 语言竞赛是浙江大学计算机学院主持的一个欢乐的竞赛。既然竞赛主旨是为了好玩，颁奖规则也就制定得很滑稽：

- 0、冠军将赢得一份“神秘大奖”（比如很巨大的一本学生研究论文集……）。
- 1、排名为素数的学生将赢得最好的奖品——小黄人玩偶！
- 2、其他人将得到巧克力。

给定比赛的最终排名以及一系列参赛者的 ID，你要给出这些参赛者应该获得的奖品。

输入格式：

输入第一行给出一个正整数 N ($\leq 10^4$)，是参赛者人数。随后 N 行给出最终排名，每行按排名顺序给出一位参赛者的 ID（4 位数字组成）。接下来给出一个正整数 K 以及 K 个需要查询的 ID。

输出格式：

对每个要查询的 ID，在一行中输出 **ID: 奖品**，其中奖品或者是 **Mystery Award**（神秘大奖）、或者是 **Minion**（小黄人）、或者是 **Chocolate**（巧克力）。如果所查 ID 根本不在排名里，打印 **Are you kidding?**（要我呢？）。如果该 ID 已经查过了（即奖品已经领过了），打印 **ID: Checked**（不能多吃多占）。

输入样例：

```
6
1111
6666
8888
1234
5555
0001
6
8888
0001
1111
2222
8888
2222
```

输出样例：

```
8888: Minion
0001: Chocolate
1111: Mystery Award
2222: Are you kidding?
8888: Checked
2222: Are you kidding?
```

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

using namespace std;

int ran[10000];
```



```

bool is_prime(int a)
{
    for(int i = 1; i * i < a ; i++)
        if(a % i == 0) return false;
    return true;
}
int main()
{
    int n;
    cin >> n;
    for(int i = 0; i < n; i++)
    {
        int id;
        cin >> id;
        ran[id] = i + 1;
    }
    int k;
    cin >> k;
    set<int> ss;
    for(int i = 0; i < k; i++)
    {
        int id;
        scanf("%d", &id);
        printf("%04d: ", id);
        if(ran[id] == 0) {
            printf("Are you kidding?\n");
            continue;
        }
        if(ss.find(id) == ss.end()) {
            ss.insert(id);
        } else {
            printf("Checked\n");
            continue;
        }
        if(ran[id] == 1) {
            printf("Mystery Award\n");
        } else if(is_prime(ran[id])) {
            printf("Minion\n");
        } else {
            printf("Chocolate\n");
        }
    }
}

```

1060 爱丁顿数 (25分)

英国天文学家爱丁顿很喜欢骑车。据说他为了炫耀自己的骑车功力，还定义了一个“爱丁顿数” E ，即满足有 E 天骑车超过 E 英里的最大整数 E 。据说爱丁顿自己的 E 等于87。

现给定某人 N 天的骑车距离，请你算出对应的爱丁顿数 E ($\leq N$)。

输入格式：



输入第一行给出一个正整数 N ($\leq 10^5$)，即连续骑车的天数；第二行给出 N 个非负整数，代表每天的骑车距离。

输出格式：

在一行中给出 N 天的爱丁顿数。

输入样例：

```
10
6 7 6 9 3 10 8 2 7 8
```

输出样例：

```
6
```

```
#include <iostream>
#include <algorithm>

using namespace std;

int a[1000000];

int cmp1(int a, int b) {
    return a > b;
}

int main() {
    int n;
    scanf("%d", &n);
    for(int i = 1; i <= n; i++) scanf("%d", &a[i]);
    sort(a+1, a+n+1, cmp1);
    int ans = 0, p = 1;
    while(ans <= n && a[p] > p) {
        ans++;
        p++;
    }
    printf("%d", ans);
    return 0;
}
```