

OUCEECPL 课程任务三

郑海永

2016 年春季

注意：“数据成分”和“运算成分”各任选三题撰写报告（其余题目自己练习）

目录

1	数据成分	1
1.1	抄写题 1：约瑟夫问题	1
1.2	抄写题 2：分数求和	3
1.3	编程题 1：年龄与疾病	5
1.4	编程题 2：成绩判断	6
1.5	编程题 3：找出第 k 大的数	7
1.6	编程题 4：人民币支付	7
2	运算成分	8
2.1	抄写题 1：点评赛车	8
2.2	编程题 1：数字求和	10
2.3	编程题 2：骑车与走路	10
2.4	编程题 3：买房子	11
2.5	编程题 4：找和为 K 的两个元素	12
2.6	编程题 5：自整除数	12

1. 数据成分

1.1 抄写题 1：约瑟夫问题

<http://oucee.openjudge.cn/a3/1>

来源

POJ 1748

注意 总时间限制: 1000ms 内存限制: 65536kB

描述 约瑟夫问题: 有 n 只猴子, 按顺时针方向围成一圈选大王 (编号从 1 到 n), 从第 1 号开始报数, 一直数到 m , 数到 m 的猴子退出圈外, 剩下的猴子再接着从 1 开始报数。就这样, 直到圈内只剩下一只猴子时, 这个猴子就是猴王, 编程求输入 n, m 后, 输出最后猴王的编号。

输入 每行是用空格分开的两个整数, 第一个是 n , 第二个是 m ($0 < m, n \leq 300$)。最后一行是:

```
1 0 0
```

输出 对于每行输入数据 (最后一行除外), 输出数据也是一行, 即最后猴王的编号。

样例输入

```
1 6 2
2 12 4
3 8 3
4 0 0
```

样例输出

```
1 5
2 1
3 7
```

参考答案

请完全按照如下的程序书写代码, 并在书写的过程中体会优秀的代码风格:

```
1 #include<iostream>
2 using namespace std;
3
4 //一共最多有 300 只猴子
5 int succedent[300]; //这个数组用于保存一个猴子后一位是谁,
6 //比如 "next[5] 的值是 7" 就是说 5 号猴子的下一位是 7 号猴子, 6 号猴子已经在之前退出了。
7 int precedent[300]; //这个数组用于保存一个猴子前一位是谁, 用法和上面的类似。
8
9 int main() {
10     int n, m;
11     while (true) {
12         cin >> n >> m;
13         if (n == 0 && m == 0)
14             break;
15         for (int i = 0; i < n - 1; i++) {
16             succedent[i] = i + 1;
17             precedent[i + 1] = i;
18         }
19         succedent[n - 1] = 0;
20         precedent[0] = n - 1;
21
22         int current = 0;
23         while (true) {
24             //如果一共要报 m 次号, 那么取 m-1 次 succedent 之后就是需要退出的那只猴子
25             for (int count = 0; count < m-1; count++)
26                 current = succedent[current];
27
28             int pre = precedent[current];
29             int suc = succedent[current];
30             //让 current 号猴子退出很简单, 就是把前一位的“下一位”指向 current 的下一位,
31             //下一位的“前一位”指向 current 的前一位就好了
32             succedent[pre] = suc;
33             precedent[suc] = pre;
34             if (pre == suc) {
35                 //如果只剩下两个了, 那么每个人的前位和后位就是同一个了。
36                 //current 是退出的, 那么另一个就是剩下的。
37                 //我们的序号是从 0 编号的, 输出时要加一
38                 cout << pre+1 << endl;
39                 break;
40             }
41             current = suc;
42         }
43     }
44     return 0;
45 }
```

1.2 抄写题 2：分数求和

<http://oucee.openjudge.cn/a3/2>

来源

POJ 6180

注意

总时间限制: 1000ms 内存限制: 65536kB

描述

输入 n 个分数并对他们求和, 用约分之后的最简形式表示。比如:

$$\frac{q}{p} = \frac{x_1}{y_1} + \frac{x_2}{y_2} + \cdots + \frac{x_n}{y_n}$$

$\frac{q}{p}$ 要求是归约之后的形式。

如： $\frac{5}{6}$ 已经是最简形式， $\frac{3}{6}$ 需要规约为 $\frac{1}{2}$ ， $\frac{3}{1}$ 需要规约成 3， $\frac{10}{3}$ 就是最简形式。

PS：分子和分母都没有为 0 的情况，也没有出现负数的情况。

输入

- 第一行的输入 n ，代表一共有几个分数需要求和；
- 接下来的 n 行是分数。

输出

输出只有一行，即归约后的结果。

样例输入

```
1 2
2 1/2
3 1/3
```

样例输出

```
1 5/6
```

参考答案

请完全按照如下的程序书写代码，并在书写的过程中体会优秀的代码风格：

```
1 #include<iostream>
2 using namespace std;
3
4 int main() {
5     int n;
6     cin >> n;
7     int sumn = 0, sumd = 1; // 储存结果, sumn/sumd
8     while (n-- > 0) {
9         int num, deno;
10        char slash; // 专门用来吃掉 / 的
11        cin >> num >> slash >> deno;
12        // 先相加  $a/b + c/d = (a*d+c*b)/(b*d)$ 
13        sumn = sumn*deno + num*sumd;
14        sumd = sumd*deno;
15    }
16    // 后约分
17    // 先求最大公约数 gcd, 这里用的是欧几里得法
18    int a = sumd, b = sumn, c;
19    while (a != 0) {
20        c = a; a = b%a; b = c;
21    }
22    int gcd = b;
23    // 分子分母同时除以 gcd 就可以完成约分
24    sumd /= gcd;
25    sumn /= gcd;
26    if (sumd > 1)
27        cout << sumn << '/' << sumd << endl;
28    else
29        cout << sumn << endl;
30    return 0;
31 }
32 // 我们计算过程中结果分母是不断乘以新输入的分母, 最后约分的。这样可能导致这个过程中分母过大溢出。
33 // 这道题的数据比较简单, 并没有出现那种情况。但大家可以思考一下, 如果出现了那种情况怎么办呢?(不要用大整数啊)
34 /* 我给大家一组测试数据, 看看你修改过的程序能不能通过这组数据吧:
35 样例输入:
36 2
37 1/100000000
38 1/100000000
39 样例输出:
40 1/500000000
41 */
```

1.3 编程题 1：年龄与疾病

<http://oucee.openjudge.cn/a3/3>

来源

POJ 6181

注意

总时间限制: 1000ms 内存限制: 65536kB

描述

某医院想统计一下某项疾病的获得与否与年龄是否有关, 需要对以前的诊断记录进行整理。

输入

共 2 行, 第一行为过往病人的数目 n ($0 < n \leq 100$), 第二行为每个病人患病时的年龄。

输出

每个年龄段 (分四段: 18 以下, 19-35, 36-60, 大于 60 注意看样例输出的格式) 的患病人数占总患病人数的比例, 以百分比的形式输出, 精确到小数点后两位 (double)。关于 C++ 的格式化的输入输出, 请参考: <http://www.cplusplus.com/reference/iomanip>。也可以在网上搜索一下, 资料很多的。

样例输入

```
1 10
2 1 11 21 31 41 51 61 71 81 91
```

样例输出

```
1 1-18: 20.00%
2 19-35: 20.00%
3 36-60: 20.00%
4 60-: 40.00%
```

提示

- 注意最后一行的输出是“60-:”，而不是“61-:”。
- 每个冒号之后有一个空格。
- 输出可以用 `cout<<fixed<<setprecision(2)<<f` 来保留 `f` 后面的两位小数。

1.4 编程题 2：成绩判断

<http://oucee.openjudge.cn/a3/4>

来源 POJ 6219

注意 总时间限制: 1000ms 内存限制: 6000kB

描述 输入一个 0-100 的分数，判断分数代表什么等级。

- $95 \leq \text{分数} \leq 100$ ，输出 1；
- $90 \leq \text{分数} < 95$ ，输出 2；
- $85 \leq \text{分数} < 90$ ，输出 3；
- $80 \leq \text{分数} < 85$ ，输出 4；
- $70 \leq \text{分数} < 80$ ，输出 5；
- $60 \leq \text{分数} < 70$ ，输出 6；
- 分数 < 60 ，输出 7。

输入 n

输出 m

样例输入

```
1 87
```

样例输出

```
1 3
```

1.5 编程题 3：找出第 k 大的数

<http://oucee.openjudge.cn/a3/5>

来源 POJ 6344

注意 总时间限制: 1000ms 内存限制: 65536kB

描述 用户输入 N 和 K ，然后接着输入 N 个正整数（无序的），程序在不对 N 个整数排序的情况下，找出第 K 大的数。注意，第 K 大的数意味着从大到小排在第 K 位的数。

输入

```
1 N
2 K
3 a1 a2 a3 a4 ... aN
```

输出

```
1 b
```

样例输入

```
1 5
2 2
3 32 3 12 5 89
```

样例输出

```
1 32
```

提示 这是一道很经典的算法问题，是公司面试的常见考题。以后学习递归之后再回头看看这道题，或许有新解法。

1.6 编程题 4：人民币支付

<http://oucee.openjudge.cn/a3/6>

来源 POJ 6183

注意 总时间限制: 1000ms 内存限制: 65536kB

描述 从键盘输入一指定金额（以元为单位，如 345），然后输出支付该金额的各种面额的人民币

数量，显示 100 元，50 元，20 元，10 元，5 元，1 元各多少张，要求尽量使用大面额的钞票。

输入 一个小于 1000 的正整数。

输出 输出分行，每行显示一个整数，从上到下分别表示 100 元，50 元，20 元，10 元，5 元，1 元人民币的张数。

样例输入

```
1 735
```

样例输出

```
1 7
2 0
3 1
4 1
5 1
6 0
```

2. 运算成分

2.1 抄写题 1：点评赛车

<http://oucee.openjudge.cn/a3/7>

来源 POJ 6536

注意 总时间限制: 1000ms 内存限制: 65536kB

描述 4 名专家对 4 款赛车进行评论

1. A 说：2 号赛车是最好的；
2. B 说：4 号赛车是最好的；
3. C 说：3 号赛车不是最好的；
4. D 说：B 说错了。

事实上只有 1 款赛车最佳，且只有 1 名专家说对了，其他 3 人都说错了。

请编程输出最佳车的车号，以及说对的专家。

输入 无输入。

输出 输出两行。第一行输出最佳车的车号（1-4 中的某个数字）。第二行输出说对的专家（A-D

中的某个字母)。

样例输入

```
1 (无)
```

样例输出

```
1 1
2 A
```

提示

样例输出只是格式说明，并非正确答案

通过这道题我们想让大家知道如何通过枚举处理逻辑判断问题。

参考答案

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     // autograder 是怎么工作的？
5     // 答案就是：autograder 只看在所有测试用例上的输出是否正确。
6     // 由于本题没有输入，输出也唯一，
7     // 所以，如果我们知道了答案（例如人工地一一枚举过来）那我们就可以直接输出啦！结果如下：
8     //     cout << "3" << endl << "D" << endl;
9     // oj 系统的 autograder 可辨识不了我们的程序使用的算法是否符合要求
10    //（事实上，任何关于程序的非凡性质都是不可判定的！
11    // ——等等，那 autograder 怎么工作呢？
12    // ——原来，autograder 限制了程序运行的时间。其目的当然包括要求我们使用的算法不能太笨拙。
13    //     更重要的是，如果我们允许最笨拙的方法并且对时间不作任何限制，
14    //     那么 autograder 就不能正确判定所有的程序啦。）
15    // 不过呢，这么输出是对 autograder 的一种错误使用方式，因为显然这不是本道题的考试内容。
16    // 所以，我们还是老老实实地用“程序”来解这个问题的答案。
17
18    // 用 best 枚举最好的车
19    int best;
20    for(best = 1; best <= 4; best++){
21        // a b c d 记录四位专家的话
22        bool a = (best == 2);
23        bool b = (best == 4);
24        bool c = !(best == 3);
25        bool d = !b;
26        if (a + b + c + d != 1)
27            continue; // 不符合只有 1 位专家说对的条件
28            // 输出最佳的车
29        cout << best << endl;
30        // 输出判断正确的专家
31        if (a == 1)
32            cout << "A" << endl;
33        else if (b == 1)
34            cout << "B" << endl;
35        else if (c == 1)
36            cout << "C" << endl;
37        else
38            cout << "D" << endl;
39    }
40    return 0;
41 }
```

2.2 编程题 1：数字求和

<http://oucee.openjudge.cn/a3/8>

来源 POJ 2682

注意 总时间限制: 1000ms 内存限制: 65536kB

描述 给定一个正整数 a ，以及另外的 5 个正整数，问题是：这 5 个整数中，小于 a 的整数的和是多少？

输入 输入一行，只包括 6 个小于 100 的正整数，其中第一个正整数就是 a 。

输出 输出一行，给出一个正整数，是 5 个数中小于 a 的数的和。

样例输入

```
1 10 1 2 3 4 11
```

样例输出

```
1 10
```

2.3 编程题 2：骑车与走路

<http://oucee.openjudge.cn/a3/9>

来源 POJ 1705

注意 总时间限制: 1000ms 内存限制: 65536kB

描述 在北大校园里，没有自行车，上课办事会很不方便。但实际上，并非去办任何事情都是骑车快，因为骑车总要找车、开锁、停车、锁车等，这要耽误一些时间。假设找到自行车，开锁并车上自行车的时间为 27 秒；停车锁车的时间为 23 秒；步行每秒行走 1.2 米，骑车每秒行走 3.0 米。请判断走不同的距离去办事，是骑车快还是走路快。

输入

- 第一行为待处理的数据的数量 n ；
- 其后每一行整数为一次办事要行走的距离，单位为米。

输出

对应每个整数，

- 如果骑车快，输出一行“Bike”；
- 如果走路快，输出一行“Walk”；

- 如果一样快，输出一行 “All”。

样例输入

```
1 4
2 50
3 90
4 120
5 180
```

样例输出

```
1 Walk
2 Walk
3 Bike
4 Bike
```

提示

注意数据类型，应当使用浮点数来保存结果。

2.4 编程题 3：买房子

<http://oucee.openjudge.cn/a3/10>

来源

POJ 3094

注意

总时间限制: 1000ms 内存限制: 65536kB

描述

某程序员开始工作，年薪 N 万，他希望在中关村公馆买一套 60 平米的房子，现在价格是 200 万，假设房子价格以每年百分之 K 增长，并且该程序员未来年薪不变，且不吃不喝，不用交税，每年所得 N 万全都积攒起来，问第几年能够买下这套房子（第一年房价 200 万，收入 N 万）。程序员每年先拿工资，再尝试买房，然后房子才涨价。

输入

有多行，每行两个整数 N ($10 \leq N \leq 50$), K ($1 \leq K \leq 20$)。

输出

针对每组数据，如果在第 20 年或者之前就能买下这套房子，则输出一个整数 M ，表示最早需要在第 M 年能买下，否则输出 Impossible，输出需要换行。

样例输入

```
1 50 10
2 40 10
3 40 8
```

样例输出

```
1 8
2 Impossible
3 10
```

提示

注意数据类型，应当使用浮点数来保存结果。

C++ 里多行输入（在不知道一共有多少行的情况下）可以使用下面的语句，每输入一组数据就可以输出其结果，不用等待所有数据都输入完毕。

```
1 while(cin>>N>>K){
2     //do your magic
3 }
```

2.5 编程题 4：找和为 K 的两个元素

<http://oucee.openjudge.cn/a3/11>

来源

POJ 6184

注意

总时间限制: 1000ms 内存限制: 65536kB

描述

在一个长度为 n ($n < 1000$) 的整数序列中，判断是否存在某两个元素之和为 k 。

输入

- 第一行输入序列的长度 n 和 k ，用空格分开。
- 第二行输入序列中的 n 个整数，用空格分开。

输出

如果存在某两个元素的和为 k ，则输出 yes，否则输出 no。

样例输入

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

样例输出

```
1 yes
```

2.6 编程题 5：自整除数

<http://oucee.openjudge.cn/a3/12>

来源 POJ 1701

注意 总时间限制: 1000ms 内存限制: 65536kB

描述 对一个整数 n ，如果其各个位数的数字相加得到的数 m 能整除 n ，则称 n 为自整除数。例如 21， $21\%(2+1) == 0$ ，所以 21 是自整除数。现求出从 10 到 n ($n < 100$) 之间的所有自整除数。

输入 有一行，整数 n ($10 \leq n < 100$)。

输出 有多行。按从小到大的顺序输出所有大于等于 10，小于等于 n 的自整除数，每行一个自整除数。

样例输入

```
1 47
```

样例输出

```
1 10
2 12
3 18
4 20
5 21
6 24
7 27
8 30
9 36
10 40
11 42
12 45
```