分糖果（candy）

【题目背景】

红太阳幼儿园的小朋友们开始分糖果啦！

【题目描述】

红太阳幼儿园有 n 个小朋友，你是其中之一。保证 $n \geq 2$。

有一天你在幼儿园的后花园里发现无穷多颗糖果，你打算拿一些糖果回去分给幼儿园的小朋友们。

由于你只是个平平无奇的幼儿园小朋友，所以你的体力有限，至多只能拿 R 块糖回去。

但是拿的太少不够分的，所以你至少要拿 L 块糖回去。保证 $n \leq L \leq R$。

也就是说，如果你拿了 k 块糖，那么你需要保证 $L \leq k \leq R$。

如果你拿了 k 块糖，你将把这 k 块糖放到篮子里，并要求大家按照如下方案分糖果：只要篮子里有**不少于** n 块糖果，幼儿园的所有 n 个小朋友（包括你自己）都从篮子中拿走**恰好**一块糖，直到篮子里的糖数量**少于** n 块。此时篮子里剩余的糖果均归你所有——这些糖果是**作为你搬糖果的奖励**。

作为幼儿园高质量小朋友，你希望让**作为你搬糖果的奖励**的糖果数量（**而不是你最后获得的总糖果数量！**）尽可能多；因此你需要写一个程序，依次输入 n, L, R，并输出出你最多能获得多少**作为你搬糖果的奖励**的糖果数量。

【输入格式】

从文件 ***candy.in*** 中读入数据。

输入一行，包含三个正整数 n, L, R，分别表示小朋友的个数、糖果数量的下界和上界。

【输出格式】

输出到文件 ***candy.out*** 中。

输出一行一个整数，表示你最多能获得的**作为你搬糖果的奖励**的糖果数量。

【样例 1 输入】

```
7 16 23
```

【样例 1 输出】

```
6
```

【样例 1 解释】

拿 $k = 20$ 块糖放入篮子里。

篮子里现在糖果数 $20 \geq n = 7$，因此所有小朋友获得一块糖；

篮子里现在糖果数变成 $13 \geq n = 7$，因此所有小朋友获得一块糖；

篮子里现在糖果数变成 $6 < n = 7$，因此这 6 块糖是作为你搬糖果的奖励。

容易发现，你获得的作为你搬糖果的奖励的糖果数量不可能超过 6 块（不然，篮子里的糖果数量最后仍然不少于 n，需要继续每个小朋友拿一块），因此答案是 6。

【样例 2 输入】

```
10 14 18
```

【样例 2 输出】

```
8
```

【样例 2 解释】

容易发现，当你拿的糖数量 k 满足 $14 = L \leq k \leq R = 18$ 时，所有小朋友获得一块糖后，剩下的 $k - 10$ 块糖总是作为你搬糖果的奖励的糖果数量，因此拿 $k = 18$ 块是最优解，答案是 8。

【样例 3】

见选手目录下的 ***candy/candy3.in*** 与 ***candy/candy3.ans***。

【数据范围】

测试点	$n \leq$	$R \leq$	$R-L \leq$
1	2	5	5
2	5	10	10
3	10^3	10^3	10^3
4	10^5	10^5	10^5
5	10^3	10^9	0
6			10^3
7	10^5		10^5
8	10^9		10^9
9			
10			

对于所有数据，保证 $2 \leq n \leq L \leq R \leq 10^9$。

插入排序（sort）

【题目描述】

插入排序是一种非常常见且简单的排序算法。小 Z 是一名大一的新生，今天 H 老师刚刚在上课的时候讲了插入排序算法。

假设比较两个元素的时间为 $O(1)$，则插入排序可以以 $O(n^2)$ 的时间复杂度完成长度为 n 的数组的排序。不妨假设这 n 个数字分别存储在 $a_1, a_2, \cdots, a_n$ 之中，则如下伪代码给出了插入排序算法的一种最简单的实现方式：

这下面是 C/C++ 的示范代码

```
for (int i = 1; i <= n; i++)
    for (int j = i; j>=2; j--)
        if ( a[j] < a[j-1] ){
            int t = a[j-1];
            a[j-1] = a[j];
            a[j] = t;
        }
```

这下面是 Pascal 的示范代码

```
for i:=1 to n do
    for j:=i downto 2 do
        if a[j]<a[j-1] then
            begin
                t:=a[i];
                a[i]:=a[j];
                a[j]:=t;
            end;
```

为了帮助小 Z 更好的理解插入排序，小 Z 的老师 H 老师留下了这么一道家庭作业：

H 老师给了一个长度为 n 的数组 a，数组下标从 1 开始，并且数组中的所有元素均为非负整数。小 Z 需要支持在数组 a 上的 Q 次操作，操作共两种，参数分别如下：

1 x v：这是第一种操作，会将 a 的第 x 个元素，也就是 a_x 的值，修改为 v。保证 $1 \leq x \leq n, 1 \leq v \leq 10^9$。**注意这种操作会改变数组的元素，修改得到的数组会被保留，也会影响后续的操作。**

2 x：这是第二种操作，假设 H 老师按照**上面的伪代码**对 a 数组进行排序，你需要告诉 H 老师原来 a 的第 x 个元素，也就是 a_x，在排序后的新数组所处的位置。保证 $1 \leq x \leq n$。**注意这种操作不会改变数组的元素，排序后的数组不会被保留，也不会影响后续的操作。**

H 老师不喜欢过多的修改，所以他保证类型 1 的操作次数不超过 5000。

小 Z 没有学过计算机竞赛，因此小 Z 并不会做这道题。他找到了你来帮助他解决这个问题。

【输入格式】

从文件 ***sort.in*** 中读入数据。

输入的第一行包含两个正整数 n, Q，表示数组长度和操作次数。保证 $1 \leq n \leq 8,000, 1 \leq Q \leq 2 \times 10^5$。

输入的第二行包含 n 个空格分隔的非负整数，其中第 i 个非负整数表示 a_i。保证 $1 \leq a_i \leq 10^9$。

接下来 Q 行，每行 $2 \sim 3$ 个正整数，表示一次操作，操作格式见题目描述。

【输出格式】

输出到文件 ***sort.out*** 中。

对于每一次类型为 2 的询问，输出一行一个正整数表示答案。

【样例 1 输入】

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

【样例 1 输出】

```
1
1
2
```

【样例 1 解释】

在修改操作之前，假设 H 老师进行了一次插入排序，则原序列的三个元素在排序结束后所处的位置分别是 $3, 2, 1$。

在修改操作之前，假设 H 老师进行了一次插入排序，则原序列的三个元素在排序结束后所处的位置分别是 $3, 1, 2$。

注意虽然此时 $a_2 = a_3$，但是我们**不能将其视为相同的元素**。

【样例 2】

见选手目录下的 ***sort/sort2.in*** 与 ***sort/sort2.ans***。

该测试点数据范围同测试点 $1 \sim 2$。

【样例 3】

见选手目录下的 ***sort/sort3.in*** 与 ***sort/sort3.ans***。

该测试点数据范围同测试点 $3 \sim 7$。

【样例 4】

见选手目录下的 ***sort/sort4.in*** 与 ***sort/sort4.ans***。

该测试点数据范围同测试点 $12 \sim 14$。

【数据范围】

对于所有测试数据，满足 $1 \leq n \leq 8,000, 1 \leq Q \leq 2 \times 10^5, 1 \leq x \leq n, 1 \leq v, a_i \leq 10^9$。

对于所有测试数据，保证在所有 Q 次操作中，至多有 5000 次操作属于类型一。

各测试点的附加限制及分值如下表所示。

测试点	n	Q	特殊性质
1,2,3,4	≤ 10	≤ 10	无
5,6,7,8,9	≤ 300	≤ 300	
10,11,12,13	$\leq 1,500$	$\leq 1,500$	
14,15,16	$\leq 8,000$	$\leq 8,000$	保证所有输入的 a_i, v 互不相同
17,18,19			无
20,21,22		$\leq 2 \times 10^5$	保证所有输入的 a_i, v 互不相同
23,24,25			无

网络连接（network）

【题目描述】

TCP/IP 协议是网络通信领域的一项重要协议。今天你的任务，就是尝试利用这个协议，还原一个简化后的网络连接场景。

在本问题中，计算机分为两大类：服务机（`Server`）和客户机（`Client`）。服务机负责建立连接，客户机负责加入连接。

需要进行网络连接的计算机共有 n 台，编号为 $1 \sim n$，这些机器将按编号递增的顺序，依次发起一条建立连接或加入连接的操作。

每台机器在尝试建立或加入连接时需要提供一个地址串。服务机提供的地址串表示它尝试建立连接的地址，客户机提供的地址串表示它尝试加入连接的地址。

一个符合规范的地址串应当具有以下特征：

1、必须形如 `a.b.c.d:e` 的格式，其中 a,b,c,d,e 均为非负整数；

2、$0 \leq a,b,c,d \leq 255, 0 \leq e \leq 65535$；

3、a,b,c,d,e 均不能含有多余的前导 0。

相应地，不符合规范的地址串可能具有以下特征：

1、不是形如 `a.b.c.d:e` 格式的字符串，例如含有多于 3 个字符 `.` 或多于 1 个字符 `:` 等情况；

2、整数 a,b,c,d,e 中某一个或多个超出上述范围；

3、整数 a,b,c,d,e 中某一个或多个含有多余的前导 0 。

例如，地址串 `192.168.0.255:80` 是符合规范的，但 `192.168.0.999:80`、`192.168.00.1:10`、`192.168.0.1:088`、`192:168:0:1.233` 均是不符合规范的。

如果服务机或客户机在发起操作时提供的地址串不符合规范，这条操作将被直接忽略。

在本问题中，我们假定凡是符合上述规范的地址串均可参与正常的连接，你无需考虑每个地址串的实际意义。

由于网络阻塞等原因，不允许两台服务机使用相同的地址串，如果此类现象发生，后一台尝试建立连接的服务机将会无法成功建立连接；除此之外，凡是提供符合规范的地址串的服务机均可成功建立连接。

如果某台提供符合规范的地址的客户机在尝试加入连接时，与先前某台已经成功建立连接的服务机提供的地址串相同，这台客户机就可以成功加入连接，并称其连接到这台服务机；如果找不到这样的服务机，则认为这台客户机无法成功加入连接。

请注意，尽管不允许两台不同的服务机使用相同的地址串，但多台客户机使用同样的地址串，以及同一台服务机同时被多台客户机连接的情况是被允许的。

你的任务很简单：在给出每台计算机的类型以及地址串之后，判断这台计算机的连接情况。

【输入格式】

从文件 ***network.in*** 中读入数据。

第 1 行，一个正整数 n 。

接下来 n 行，每行 2 个字符串 op, ad ，按照编号从小到大给出每台计算机的类型及地址串。

其中 op 保证为字符串 `Server` 或 `Client` 之一，ad 为一个长度不超过 25 的，仅由数字、字符 `.` 和字符 `:` 组成的非空字符串。

每行的两个字符串之间用恰好一个空格分隔开，每行的末尾没有多余的空格。

【输出格式】

输出到文件 ***network.out*** 中。

输出共 n 行，每行一个正整数或字符串表示第 i 台计算机的连接状态。其中：

如果第 i 台计算机为服务机，则：

1. 如果其提供符合规范的地址串且成功建立连接，输出字符串 `OK` 。
2. 如果其提供符合规范的地址串，但由于先前有相同地址串的服务机而无法成功建立连接，输出字符串 `FAIL` 。
3. 如果其提供的地址串不是符合规范的地址串，输出字符串 `ERR` 。

如果第 i 台计算机为客户机，则：

1. 如果其提供符合规范的地址串且成功加入连接，输出一个正整数表示这台客户机连接到的服务机的编号。
2. 如果其提供符合规范的地址串，但无法成功加入连接时，输出字符串 `FAIL` 。
3. 如果其提供的地址串不是符合规范的地址串，输出字符串 `ERR` 。

【样例 1 输入】

```
5
Server 192.168.1.1:8080
Server 192.168.1.1:8080
Client 192.168.1.1:8080
Client 192.168.1.1:80
Client 192.168.1.1:99999
```

【样例 1 输出】

```
OK
FAIL
```

```
1
FAIL
ERR
```

【样例 1 解释】

计算机 1 为服务机，提供符合规范的地址串 192.168.1.1:8080 ，成功建立连接；

计算机 2 为服务机，提供与计算机 1 相同的地址串，未能成功建立连接；

计算机 3 为客户机，提供符合规范的地址串 192.168.1.1:8080 ，成功加入连接，并连接到服务机 1；

计算机 4 为客户机，提供符合规范的地址串 192.168.1.1:80 ，找不到服务机与其连接；

计算机 5 为客户机，提供的地址串 192.168.1.1:99999 不符合规范。

【样例 2 输入】

```
10
Server 192.168.1.1:80
Client 192.168.1.1:80
Client 192.168.1.1:8080
Server 192.168.1.1:80
Server 192.168.1.1:8080
Server 192.168.1.999:0
Client 192.168.1.1.8080
Client 192.168.1.1:8080
Client 192.168.1.1:80
Client 192.168.1.999:0
```

【样例 2 输出】

```
OK
1
FAIL
FAIL
OK
ERR
ERR
```

```
5
1
ERR
```

【样例 3】

见选手目录下的 ***network/network3.in*** 与 ***network/network3.ans***。

【样例 4】

见选手目录下的 ***network/network4.in*** 与 ***network/network4.ans***。

【数据范围】

测试点编号	$n \leq$	特殊性质
1	10	性质 1 2 3
2 ∼ 3	100	性质 1 2 3
4 ∼ 5	1000	
6 ∼ 8		性质 1 2
9 ∼ 11		性质 1
12 ∼ 13		性质 2
14 ∼ 15		性质 4
16 ∼ 17		性质 5
18 ∼ 20		无特殊性质

“性质 1 ”为：保证所有的地址串均符合规范；

“性质 2 ”为：保证对于任意两台不同的计算机，如果它们同为服务机或者同为客户机，则它们提供的地址串一定不同；

“性质 3 ”为：保证任意一台服务机的编号都小于所有的客户机；

“性质 4 ”为：保证所有的地址串均形如 `a.b.c.d:e` 的格式，其中 a, b, c, d, e 均为不超过 10^9 且不含有多余前导 0 的非负整数；

“性质 5 ”为：保证所有的地址串均形如 `a.b.c.d:e` 的格式，其中 a, b, c, d, e 均为只含有数字的非空字符串。

对于 100% 的数据，保证 $1 \leq n \leq 1000$ 。

小熊的果篮（fruit）

【题目描述】

小熊的水果店里摆放着一排 n 个水果。每个水果只可能是苹果或桔子，从左到右依次用正整数 1、2、3、……、n 编号。连续排在一起的同一种水果称为一个“块”。小熊要把这一排水果挑到若干个果篮里，具体方法是：每次都把每一个“块”中最左边的水果同时挑出，组成一个果篮。重复这一操作，直至水果用完。注意，每次挑完一个果篮后，“块”可能会发生变化。比如两个苹果“块”之间的唯一桔子被挑走后，两个苹果“块”就变成了一个“块”。请帮小熊计算每个果篮里包含的水果。

【输入格式】

从文件 ***fruit.in*** 中读入数据。

输入的第一行包含一个正整数 n，表示水果的数量。

输入的第二行包含 n 个空格分隔的整数，其中第 i 个数表示编号为 i 的水果的种类，1 代表苹果，0 代表桔子。

【输出格式】

输出到文件 ***fruit.out*** 中。

输出若干行。

第 i 行表示第 i 次挑出的水果组成的果篮。从小到大排序输出该果篮中所有水果的编号，每两个编号之间用一个空格分隔。

【样例 1 输入】

```
12
1 1 0 0 1 1 1 0 1 1 0 0
```

【样例 1 输出】

```
1 3 5 8 9 11
2 4 6 12
7
10
```

【样例 1 解释】

这是第一组数据的样例说明。
所有水果一开始的情况是 `1 1 0 0 1 1 1 0 1 1 0 0`，一共有 6 个块。
在第一次挑水果组成果篮的过程中，编号为 1 3 5 8 9 11 的水果被挑了出来。
之后剩下的水果是 `1 0 1 1 1 0`，一共 4 个块。
在第二次挑水果组成果篮的过程中，编号为 2 4 6 12 的水果被挑了出来。
之后剩下的水果是 `1 1`，只有 1 个块。
在第三次挑水果组成果篮的过程中，编号为 7 的水果被挑了出来。
最后剩下的水果是 `1`，只有 1 个块。
在第四次挑水果组成果篮的过程中，编号为 10 的水果被挑了出来。

【样例 2 输入】

```
20
1 1 1 1 0 0 0 1 1 1 0 0 1 0 1 1 0 0 0 0
```

【样例 2 输出】

```
1 5 8 11 13 14 15 17
2 6 9 12 16 18
3 7 10 19
4 20
```

【样例 3】

见选手目录下的 ***fruit/fruit3.in*** 与 ***fruit/fruit3.ans***。

【数据范围】

对于 10% 的数据，$n \leq 5$。
对于 30% 的数据，$n \leq 1000$。
对于 70% 的数据，$n \leq 50000$。
对于 100% 的数据，$n \leq 2 \times 10^5$。

【提示】

由于数据规模较大，建议 C/C++ 选手使用 scanf 和 printf 语句输入、输出。