问题 A: 磁铁

疯狂的科学家迈克通过排列一排排的多米诺骨牌来消遣自己。不过，他不需要多米诺骨牌，而是使用矩形磁铁。每个磁铁都有两极，正极和负极。如果把两块磁铁近距离放在一起，那么同极就会互相排斥，异极就会相互吸引。

迈克先把一块磁铁水平地放在桌子上。在接下来的每一步中，迈克在这一行的右端水平增加了一个磁铁。根据迈克把磁铁放在桌子上的方式，它要么被前一块磁铁吸引(形成一组连接在一起的多个磁铁)，要么被前一块磁铁排斥(然后迈克把这块磁铁放在与前一块磁铁相对的右边一段距离)。我们假设一个单独的磁体不与其他磁体相连，形成一个自己的磁体群。

迈克把多个磁铁排列成一排。确定磁体形成的磁体群的个数。

输入格式

输入的第一行为整数 n(1≤n≤100000)−磁铁个数。然后是 n 行。第 i 行(1≤i≤n)要么包含字符“01”(如果迈克把第 i 个磁铁放在“正负”位置)，要么包含字符“10”(如果迈克把第 i 个磁铁放在“负正”位置)。

输出格式

输出磁体形成的磁体群的个数。

输入样例 1：

6

10

10

10

01

10

10

输出样例 1：

3

样例 1 解释

测试用例有三组，由三个(10 10 10)、一个(01)和两个磁铁组成(10 10)。

输入样例 2：

4

01

01

10

10

输出样例 2：

2

样例 2 解释

第二个测试用例有两组，每组由两个磁铁组成。

题解：

根据磁铁的性质，结合题目中用 10 和 01 来模拟磁铁，那么只有 10 10 或者 01 01 这样的情况才会出现相吸的情况并且结合成一个磁体群。那么我们定义一个 f 来判记录上一个磁铁的摆放方式。

代码如下：

```
#include<bits/stdc++.h>
using namespace std;
const int N = 100010;
string s;
int ans;
int main(){
    int n;
    cin>>n;
    char f = '2';//定义成不是 1 和 0 的数字，防结果为 1 的情况
    for(int i=1;i<=n;i++)
    {
        cin>>s;
```

```
            if(f!=s[0]) //如果不能组成一个磁铁群
            {
                ans++;
                f=s[0];
            }
        }
        cout<<ans<<endl;
        return 0;
}
```

问题 B: 共素数数组

如果相邻两个元素互质，则这两个元素共素数。如果一个数组的任意两个相邻数都是共素数，那么这个数组就是共素数。

给定一个包含 n 个元素的数组，在数组各元素之间可以随意插入任何小于 1e9 的正整数，每插入一个数算一步，求令这个数列相邻元素互质的最少的操作步数。

在每次移动中，您可以在数组的任何位置插入任何不大于 1e9 的正整数。

输入格式

第一行包含整数 n (1≤n≤1000) –给定数组中的元素数量。

第二行包含 n 个整数 ai (1≤ai≤109) –数组 a 的元素。

输出格式

在第一行输出一个整数 k，表示添加到数组 a 以使其成为共素数所需的最少元素个数。

第二行应该包含 n+k 个整数 aj –数组 a 的元素。注意，新数组应该是共同素数，因此任何两个相邻的值都应该是共同素数。另外，新数组应该通过添加 k 个 1，形成共素数数组，并输出该共素数数组。

Example

Input

3

2 7 28

Output

1

2 7 1 28

Input
6
2 4 8 19 25 5
Output 3
2 1 4 1 8 19 25 1 5

[样例解释]
第一个样例中 7 和 28 不是互质关系，所以需要在它们之间插入 1
第二个样例中 2 和 4，4 和 8，25 和 5 都不是互质关系

题解：找到所有数组元素中不是互质关系（这里用 gcd 函数比较方便快捷），在他们之间加上 1，然后记录下这一操作的次数，最后输出即可

代码如下：

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1010;
int a[N];
int b[N],cnt;//储存结果的数组 b,cnt 表示元素的数量
int gcd(int a,int b)//gcd 函数模板
{
    if(a%b==0) return b;
    return gcd(b,a%b);
}
int main(){
    int n;
    cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    b[++cnt] = a[1];//一会循环从 2 开始，先把 a[1]存到 b 数组中
    for(int i=2;i<=n;i++)
    {
        if(gcd(a[i],a[i-1]) == 1) //如果是互质关系
        {
            b[++cnt] = a[i];
        }
        else//如果是非互质关系
        {
            b[++cnt] = 1;//两数之间插入 1
            b[++cnt] = a[i];
        }
    }
    cout<<cnt-n<<endl;//cnt-n 即为插入 1 的数量
    for(int i=1;i<=cnt;i++) cout<<b[i]<<" ";
    return 0;
}
```

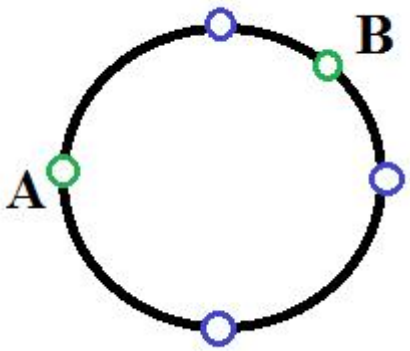
问题 C: 大傻和朋友

在大傻居住的小镇，在环形赛道上用障碍物跑步非常流行，所以难怪她在去上课的路上看到了以下情况：

赛道是长度为 L 的圆圈，在不同的点上有 n 个障碍物。如果你从上面看他，运动员总是以逆时针方向跑赛道。所有障碍物沿轨道彼此相距整数。

她的朋友鹦鹉 Kefa 和豹子 Sasha 参加了比赛，他们每个人都跑了一圈。每个朋友都从赛道上的某个积分点开始。两个朋友都写下了从他们沿着轨道开始到 n 个障碍中的每个障碍的距离。

因此，他们每个人都按升序写了 n 个整数，每个整数都在 0 和 L-1 之间，包括 0 和 L-1。



考虑一个例子。设 L=8，蓝点是障碍，绿点是 Kefa 的起点（A）和萨沙的起点（B）。然后凯法记下序列[2，4，6]，萨沙写下[1，5，7]。

小镇有几条轨道，它们都具有相同的长度和相同数量的障碍物，但障碍物的位置在不同的轨道之间可能会有所不同。现在大傻很感兴趣，如果 Kefa 和 Sasha 可能跑同一条赛道，或者他们在不同的赛道上参加。

编写程序，检查 Kefa 和 Sasha 的轨道是否重合（这意味着可以通过更改起始位置从另一个获得一个）。请注意，它们总是沿一个方向运行轨道 - 逆时针方向，如果您从上方看轨道。

输入格式

输入

第一行包含两个整数 n 和 L （1≤n≤50， n≤L≤100） - 轨道上的障碍数及其长度。

第二行包含 n 个按升序排列的不同整数 - 从 Kefa 的起点到每个障碍的距离，按其出现的顺序排列。所有整数都在 0 到 L-1 的范围内（包括 0 和 L-1）。

第二行包含 n 个按升序排列的不同整数 - 从萨沙开始到每个障碍的距离，按其克服的顺序排列。所有整数都在 0 到 L-1 的范围内（包括 0 和 L-1）。

输出格式

输出

打印“YES”（不带引号），如果 Kefa 和 Sasha 运行重合的轨道（这意味着如果它们从轨道上的相同点开始运行，则所有障碍物的位置重合）。否则打印“NO”（不带引号）。

input
3 8
2 4 6
1 5 7
output
YES
input
4 9
2 3 5 8
0 1 3 6
output
YES
input
2 4
1 3
1 2
output
NO

题目大意：给定两个环，判断它们是否相同。此题数据量并不大 因此可以使用暴力的方式：通过旋转 A 环，每次将环上的第 i 个元素移到第 i+1 个位置(最后一个元素移到第一个元素的位置），并将旋转完毕后的 A 环与 B 环相比较，即可得出正解。本题可以采用 取余的方式处理，也可以采用化环成链的方式处理，方法有多种。

代码如下：

```
#include<bits/stdc++.h>
using namespace std;
int a[55],b[55];
int aa[120],bb[120];
int main()
{
    int n,L;
    cin>>n>>L;
    for(int i=1;i<=n;i++) cin>>a[i];
    for(int i=1;i<=n;i++) cin>>b[i];
```

```
for(int i=2;i<=n;i++)//计算一下障碍物之间的距离
{
	aa[i-1] = a[i] - a[i-1];//化环成链
	bb[i-1] = b[i] - b[i-1];
}
aa[n] = (L- a[n]+a[1]);//特殊处理一下
bb[n] = (L-b[n]+b[1]);
for(int i=n+1;i<=2*n;i++)
{
	aa[i] = aa[i-n];
	bb[i] = bb[i-n];
}
int flag = 0;
for(int i=1;i<=n;i++)
{
	if(aa[1]==bb[i])//找到有希望配对的起点，开始比较
	{
		int p = 1;
		for(int j=i+1,k=2;j<=i+n;j++,k++)
		{
			if(aa[k]!=bb[j])//距离不匹配，跳过本次
			{
				p = 0;
				break;
			}
		}
		if(p==1)//本次匹配成功
		{
			flag = 1;
			break;
```

```
                }
            }
        }
        if(flag)puts("YES");
        else puts("NO");
       return 0;
}
```

问题 D: 奇数和

题目描述

给定长度为 n 的整数序列 a1,a2,...,an。你的任务是在所有这样的子序列中找到这样的子序列，它的和是奇数且最大。保证给定序列包含和为奇数的子序列。

子序列是可以通过删除一些元素而不改变其余元素的顺序从另一个序列导出的序列。

您应该编写一个程序来找到最佳子序列的总和。

输入格式

第一行包含整数 n (1≤n≤105)。

第二行包含 n 个整数 a1,a2,...,an (-104≤ai≤104)。该序列至少包含一个和为奇数的子序列。

输出格式

输出结果子序列的总和。

输入

4

-2 2 -3 1

输出

3

输入

3

2 -5 -3

输出

-1

注意

在第一个样例中第二个和第四个元素之和为 3

题解：

要使得结果为奇数。还要最大

对于数组一个元素：如果是偶数，而且大于 0，可以加入最终的序列

如果是奇数，先把最大的那个加起来，以保证结果为奇数，然后为了保证结果为奇数，我们取每两个奇数为一对，如果加起来大于等于 0，我们也同样加入最终的序列。

代码如下：

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+10;
int a[N];
int ji[N],cnt;//数组奇数元素的数组 ji，元素数量 cnt
int sumou;//最大偶数子序列和
int main()
{
    int n;
    cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    for(int i=1;i<=n;i++)
    {
```

```
        if(a[i]&1)    ji[++cnt] = a[i];//是奇数就加到 ji 数组中
        else if(a[i]>=0) sumou += a[i]; //偶数且非负数就加到偶数中
    }
    sort(ji+1,ji+1+cnt);//给 ji 数组排个序
    int ans = sumou + ji[cnt];
    for(int i=cnt-1;i>=2;i--) //每两个奇数为一对，且和大于等于 0，择加入结果中
    {
        if(ji[i] + ji[i-1] >=0 )
        {
            ans += ji[i] + ji[i-1];
            i--;
        }
    }
    cout<<ans<<endl;
    return 0;
}
```