问题 A: 熊熊与比赛

熊熊 Limak 喜欢看电视上的体育节目。他今天要去看一场比赛。这场比赛持续 90 分钟，没有休息时间。

每分钟可以是有趣的，也可以是无聊的。如果连续 15 分钟是无聊的，那么 Limak 就会立即关掉电视。

现在你知道会有 n 个有趣的分钟 t1,t2,...,tn。你的任务是计算 Limak 会看多少分钟的比赛。

输入格式

输入的第一行包含一个整数 n（1≤n≤90）--有趣的分钟数。

第二行包含 n 个整数 t1,t2,...,tn（1≤t1<t2<...tn≤90），按递增顺序给出。输出格式

输出格式

打印 Limak 将观看比赛的分钟数。

输入样例 1：

3

7 20 88

输出样例 1：

35

样例 1 解释

在第一个样例中，第 21,22,...,35 分钟都很无聊，因此 Limak 会在第 35 分钟后立即关闭电视。因此，他将看 35 分钟的比赛。

输入样例 2：

9

16 20 30 40 50 60 70 80 90

输出样例 2：

15

样例 2 解释

在第二个样例中，前 15 分钟是无聊的。

输入样例 3：

9

15 20 30 40 50 60 70 80 90

输出样例 3：

90

样例 3 解释

在第三个样例中，没有连续的 15 分钟是无聊的。所以，Limak 会看完整场比赛。

题解：

一道模拟题，给出感兴趣的时刻，若不感兴趣的时间段>=15,则关电视

注意：最后一个时刻后还可以再看 15 分钟，还有边界 90。

代码如下：

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n,a[10000];
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        cin>>a[i];
    }
    for(int i=2;i<=n;i++)
    {
        if(a[i]-a[i-1]>15){
            cout<<a[i-1]+15;
            return 0;
        }
    }
    cout<<"90";
    return 0;
 }
```

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

int main()
{
    int n,a[100];
    while(~scanf("%d",&n))
    {
        for(int i=0;i<n;i++)
            scanf("%d",&a[i]);
        if(a[0]>15)
        {
            printf("15\n");
            continue;
        }
        else
        {
            int maxn=a[0];
            for(int i=1;i<n;i++)
```

```
            {
                if((a[i]-a[i-1]>15))
                {
                    break;
                }
                else
                {
                    maxn=a[i];
                 }
            }
            maxn+=15;//注意咯.....
            if(maxn>=90)
                maxn=90;
            printf("%d\n",maxn);
        }
    }
     return 0;
}
```

问题 B: 二进制数

zz 非常喜欢数学。这就是为什么当他感到无聊时，他会玩一些数字来执行一些操作。zz 得到一个二进制数 x，并希望将这个数字变成 1。当 x 不等于 1 时，zz 重复以下操作：如果 x 是奇数，则他将 x 加 1 ，否则他将 x 除以 2。zz 知道，对于任何正整数，该过程都会在有限的时间内结束。zz 应该执行多少个次才能将 x 变成 1？

输入格式
第一行包含二进制系统中的正整数 x。保证 x 的第一个数字与零不同，其位数不超过 10^6。
输出格式
打印所需数量的操作。

输入样例 1：
1
输出样例 1：
0

输入样例 2：
1001001
输出样例 2：
12

输入样例 3：
101110
输出样例 3：
8

样例解释：

让我们考虑第三个样本。数字 101110 是偶数，这意味着我们应该将其除以 2。除法后，方吉得到一个奇数 10111，并加一。数字 11000 可以连续三次除以 2，得到数字 11。剩下的就是将数字增加 1（我们得到 100），然后连续两次将其除以 2。结果，我们得到 1。

题解：

对于一个二进制数，+1 是若最后一个数是 1 ，则变为 0 ，若为 0 ，则变为 1 ，直到遇到第一个 0 。而 /2 是将最后一个 0 去除。
判断一个二进制数奇偶性，若最后一位为 1 ，则为奇数，反之为偶数。
所以可以从最后一位往前枚举，发现 0 可以直接 /2 而发现 1 要先 +1 再 /2 。
可以很容易的发现，遇到第一个 1 后，后面就不存在纯粹的遇见 0 然后 /2 ，所以这是个临界点。
并且在倒数第二位时如果加上了 1 且倒数第二位是 1 ，那么会凭空多出一位，所以要特殊判断一次。

```
#include<bits/stdc++.h>
using namespace std;
string a;
int main(){
    int n,t=0,sum=0,n1=1,i;
    getline(cin,a);
    n=a.size();
    for(i=n-1;i>=1;i--){
        if(a[i]=='1'){
            a[i-1]=a[i-1]+1;
            t=t+2;//+1,/2 两次操作
        }
        else if(a[i]=='2'){
            t=t+1; //  /2
            a[i-1]=a[i-1]+1;//进一位
        }
        else if(a[i]=='0')t++;//  /2
    }
    if(a[0]=='2')t++;//特判
    cout<<t;
}
```

问题 C: 资格赛

很快，伯兰德将举办学校团队编程奥林匹克竞赛。来自每个伯兰地区的两个团队被邀请参加奥林匹克运动会。举行了组队资格赛，有 n 名伯兰学生参加。伯兰的每个地区至少有两名男生参加。资格赛每位参赛者的结果是 0 到 800（含）的整数分数。

每个地区的团队由该地区资格赛的两名此类成员组成，他们中的任何一个都不能被同一地区的小学生取代，不包括在团队中并且获得更多积分。可能存在某些地区的团队无法唯一组建的情况，即满足上述属性的学校团队不止一个。在这种情况下，该地区需要进行额外的竞争。如果至少有一个小学生被包括在一支球队中而不包括在另一支球队中，则该地区的两支球队被认为是不同的。保证每个地区至少有两名代表参加资格赛。

根据资格赛的结果，您的任务是确定每个地区的团队，或宣布在该地区组建团队需要额外的比赛。

输入格式

输入的第一行包含两个整数 n 和 m （2≤n≤100000， 1≤m≤10000， n≥2 m）– 资格赛的参赛人数和伯兰地区的地区数量。

接下来的 n 行包含资格赛参赛者的说明，格式如下：姓氏（长度从 1 到 10 个字符的字符串，由大大小小的英文字母组成），地区编号（从 1 到 m 的整数）和参赛者的得分数（从 0 到 800 的整数， 包括）。

保证所有参与者的所有姓氏都是不同的，并且每个 m 个区域至少有两人参加。仅在字母大小写中不同的姓氏应被视为不同。

输出格式

打印 m 行。

在第 i 行上打印第 i 个区域的团队 - 以任意顺序打印两个团队成员的姓氏，或单个字符“？”（不含引号）如果您需要在该地区花费更多的资格赛。

例子

输入样例 1：

```
5 2
Ivanov 1 763
Andreev 2 800
Petrov 1 595
Sidorov 1 790
Semenov 2 503
```

输出样例 1：

```
Sidorov Ivanov
Andreev Semenov
```

样例解释 1：

在第一个样本区域中，团队是唯一确定的。

输入样例 2：

```
5 2
Ivanov 1 800
```

Andreev 2 763
Petrov 1 800
Sidorov 1 800
Semenov 2 503
输出样例 2：
?
Andreev Semenov
样例解释 2：
在第二个样本中，来自区域 2 的团队是唯一确定的，来自区域 1 的团队可以有三个团队：“Petrov”-“Sidorov”、“Ivanov”-“Sidorov”、“Ivanov”-“Petrov”，因此不可能唯一地确定团队。

题意：

给定地区及来自相应地区的人的分数，每个地区选两个分数最高的人 参加区域赛，如果选出的两个人唯一，则输出名字，否则如果还需要进行下一次比赛，输出“？”。即给出 n 个人分别 m 个组，求每个组的最大值和次大值。

特别的，当第二大和第三大相等时，输出问号。

题解：

不唯一的情况就是第二个人和第三个人的分数相同。

按地区和分数排序,再判断第二名和第三名的分数。

代码如下：

```
#include <bits/stdc++.h>
using namespace std;
const int N=2e5+4;
struct node
{
    string str;
    int pos,num;
};
node po[N];
int cmp(node x,node y)
{
    if(x.pos==y.pos)return x.num>y.num;
```

```
    return x.pos<y.pos;
}int n,m;int main()
{
    scanf("%d%d",&n,&m);
    for(int i=0;i<n;i++)
    {
        cin>>po[i].str>>po[i].pos>>po[i].num;
    }
    sort(po,po+n,cmp);
    if(po[1].pos==po[2].pos)
    {
        if(po[1].num==po[2].num)cout<<"?"<<"\n";
        else cout<<po[0].str<<" "<<po[1].str<<"\n";
    }
    else
    {
        cout<<po[0].str<<" "<<po[1].str<<"\n";
    }
    po[n].pos=po[n-1].pos;
    po[n].num=-1;
    for(int i=2;i<n-1;i++)
    {
        if(po[i].pos==po[i-1].pos)continue;
        if(po[i+1].pos==po[i+2].pos)
        {
            if(po[i+1].num==po[i+2].num)cout<<"?"<<"\n";
            else cout<<po[i].str<<" "<<po[i+1].str<<"\n";
        }
        else cout<<po[i].str<<" "<<po[i+1].str<<"\n";
    }
```

```
    return 0;
}
```

问题 D: 电子表格

在课程中，zz 使用一个著名的电子表格 excel 学习如何编辑表格。

现在 zz 有一个填满整数的表格，该表由 n 行和 m 列组成。用 $a_{i,j}$，表示位于第 i 行和第 j 列的整数。我们说，对于从 1 到 n-1 的所有 i，如果 $a_{i,j} \le a_{i+1,j}$，则表在 j 列中按非降序排序。

老师给 zz 布置了 k 个任务。对于每个任务，给出两个整数 l 和 r，zz 必须回答以下问题：如果保留从 l 到 r 的行并删除所有其他行，表格是否会在至少一列中按非降序排序？

即是否存在这样的 j， 对于从 l 到 r-1 的所有 i，使 $a_{i,j} \le a_{i+1,j}$。

请你帮忙！

输入格式

第一行为两个数 n,m($1 \le n*m \le 1000000$)

下面 n 行，每行 m 个数($1 \le a_{i,j} \le 10^9$).

下一行为 k 的值($1 \le k \le 100000$)

接下来 k 行，每行两个数 l,r(($1 \le l_i \le r_i \le n$))

输出格式

如果第 l 行到 r 行至少有一列是非递减的，则输出 Yes，否则输出 No.

输入样例：

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

输出样例：

```
Yes
```

No
Yes
Yes
Yes
No
注意：
在示例中，整个表没有在任何列中进行排序。但是，第 1-3 行在第 1 列中排序，而第 4-5 行在第 3 列中排序。

题意：

给你一个 n * m 的矩阵（n*m <= 1e5）,然后给你 q 次询问，每次询问会给你两个数字 L 和 R ，问你在第 L 行到第 R 行中（无视其他未被选中的行的数据），是否存在一列是单调不减的。

思路

暴力会 TLE

要问的是行区间 L 到 R 是否存在一列单调不减，意味着从 L 行开始有那么一列，它能单调不减的延伸到至少第 R 行，如果我们能够获得这样一个数列 c[i]，c[i] 表示第 i 行所有的列中能够向下延伸的最大的长度，那么对于每次询问，我们都能 O（1）得到答案了。

如何获取 c[i]数组，做法是枚举每一列，对于每一列，我枚举行数从下往上看，获得每一行的最大延伸长度，对于同一行的，取最大值即可获得 c[i]，最后通过 L+c[i]是否>=R 即可判断。

代码如下：

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 1e5 + 5;
int c[maxn];//行最大延伸
int main()
{
    int n, m;
    cin >> n >> m;
    int a[n + 5][m + 5];
    for(int i = 1; i <= n; i++)
```

```
        for(int j = 1; j <= m; j++)
            cin >> a[i][j];

    for(int i = 1; i <= m;i++){
        a[0][i] = 0;
        a[n + 1][i] = 0;
    }

    int cnt;
    for(int j = 1; j <= m; j++)//枚举列
    {
        cnt = 0;
        for(int i = n; i >= 1; i--)//从下往上枚举行
        {
            if(a[i][j] < a[i - 1][j])
            {
                cnt = 0;
                a[i][j] = 0;
            }
            else
                a[i][j] = ++cnt;//cnt 为延伸长度

            c[i] = max(c[i], a[i + 1][j]);//a[i+1][j]存储 a[i][j]的延长距离
        }
    }

    int q, l, r;
    cin >> q;
    while(q--)
    {
        cin >> l >> r;
        if(c[l] + l >= r)
            printf("Yes\n");
        else
            printf("No\n");
    }
    return 0;
}
```

问题 E: 悠闲的漫步

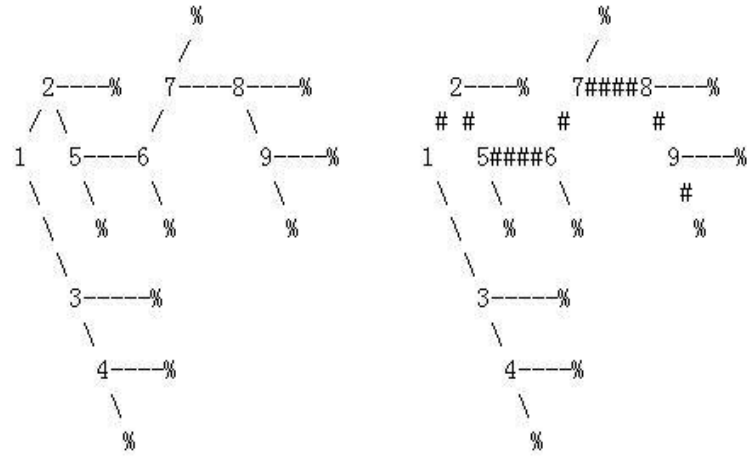
Bessie 透过牛棚的大门向外望去。发现今天是一个美丽的春季早晨。她想，“我真的好想好想沐浴著春风，走在草地之中，感受嫩草温柔地抚摸四蹄地的感觉。”

她知道一旦她离开了牛棚，她将沿著一条小径走一段路，然后就会出现一个三岔路口，

她必须在两条小径中选择一条继续走下去。然后她又会遇到更多的三岔路口，进行更多的选择，直到她到达一个青翠的牧场为止。

做一个选择使得她在去吃草的路途中可以走过最多的小径。给出小径的描述，求 Bessie 最多可以走过多少条小径。假定 Bessie 一出牛棚就有 2 条路径，Bessie 需要从中选择一条。

农场中有 P-1 (1 < = P < = 1000) 个分岔节点（范围是 1..P），引向 P 片草地，它们之间由小径连接。对任意一个节点来说，只有一条从牛棚（被标记為节点 1）开始的路径可以到达。考虑下面的图。线段表示小径，" %" 表示草地。右边的图中的" #" 表示一条到达草地的高亮的路径。



从分岔节点 9 到达的草地是两个可以让 Bessie 走过最多小径的草地之一。在去吃早草的路上 Bessie 将走过 7 条不同的小径。这些草地是离牛棚也就是节点 1 最“远”的。
由 3 个整数来表示每一个节点：Cn, D1 和 D2，Cn 是节点的编号(1 < = Cn < = P-1);
D1 和 D2 是由该节点引出的两条小径的终点(0 < = D1 < = P-1; 0 < = D2 < = P-1)。
如果 D1 为 0，表示这条小径引向的是一片牧草地；D2 也一样。

输入：
第 1 行: 一个单独的整数: P
第 2 到第 P 行: 第 i+1 行有 3 个由空格隔开的整数，表示一个分岔节点 Cn, D1 和 D2。

输出：
第一行: 一个单独的整数，表示 Bessie 去最远的草地的路上最多可以走过的小径的数目。

输入样例：
10
7 8 0
5 0 6
9 0 0
6 0 7
3 4 0
2 5 0
8 0 9
4 0 0

1 2 3

输出样例：
7

题解：
搜索就可以过，仔细思考一下这个题可以发现就是一棵树从根节点到他的任意一个子节点的最长路径。

```
#include<bits/stdc++.h>
using namespace std;
#define maxn 1005
int l[maxn], r[maxn], x, maxx = 0;
void dfs(int xx, int sum){
	if (l[xx] != 0)dfs(l[xx], sum + 1);//没到终点，往左岔路搜索
	if (r[xx] != 0)dfs(r[xx], sum + 1);//没到终点，往右岔路搜索
	maxx = max(maxx, sum);//每次刷新记录下最大值
}
int main(){
	int n;
	cin >> n;
	for (int i = 1; i < n; i++)
	{
		cin >> x;
		cin >> l[x] >> r[x];//每个岔路点通向的地方
	}
	dfs(1, 1);//深度搜索
	cout << maxx << endl;
}
```