# P1563 [NOIP2016 提高组] 玩具谜题

## 题目描述

小南有一套可爱的玩具小人，它们各有不同的职业。

有一天，这些玩具小人把小南的眼镜藏了起来。 小南发现玩具小人们围成了一个圈,它们有的面朝圈内,有的面朝圈外。如下图:


这时singersinger告诉小南一个谜题：“眼镜藏在我左数第3个玩具小人的右数第1个玩具小人的左数第2个玩具小人那里。 ”

小南发现，这个谜题中玩具小人的朝向非常关键，因为朝内和朝外的玩具小人的左右方向是相反的：面朝圈内的玩具小人，它的左边是顺时针方向，右边是逆时针方向；而面向圈外的玩具小人，它的左边是逆时针方向，右边是顺时针方向。

小南一边艰难地辨认着玩具小人，一边数着：

singersinger朝内，左数第3个是archerarcher。

archerarcher朝外,右数第1个是thinkerthinker。

thinkerthinker朝外，左数第2个是writewriter。

所以眼镜藏在writerwriter这里!

虽然成功找回了眼镜，但小南并没有放心。 如果下次有更多的玩具小人藏他的眼镜，或是谜题的长度更长，他可能就无法找到眼镜了 。 所以小南希望你写程序帮他解决类似的谜题。 这样的谜题具体可以描述为:

有 n个玩具小人围成一圈，已知它们的职业和朝向。现在第1个玩具小人告诉小南一个包含mm条指令的谜题，其中第 z条指令形如“左数/右数第s,个玩具小人”。 你需要输出依次数完这些指令后,到达的玩具小人的职业。

## 输入格式

输入的第一行包含两个正整数 n,m，表示玩具小人的个数和指令的条数。

接下来 n 行，每行包含一个整数和一个字符串，以逆时针为顺序给出每个玩具小人的朝向和职业。其中 0 表示朝向圈内，1 表示朝向圈外。 保证不会出现其他的数。字符串长度不超过 10 且仅由小写字母构成，字符串不为空，并且字符串两两不同。整数和字符串之间用一个空格隔开。

接下来 m 行，其中第 i 行包含两个整数 $a_i$，$s_i$ ，表示第 i 条指令。若$a_i$=0，表示向左数$s_i$个人；若 $a_i$=1，表示向右数 $s_i$个人。 保证 $a_i$不会出现其他的数，$1 \leqslant s_i < n$。

**输出格式**

输出一个字符串，表示从第一个读入的小人开始，依次数完 m 条指令后到达的小人的职业。

**输入样例**

```
7 3
0 singer
0 reader
0 mengbier
1 thinker
1 archer
0 writer
1 mogician
0 3
1 1
0 2
```

**输出样例**

```
writer
```

**解析**

本题的核心在于玩具处于不同朝向时如何计算它的下一个玩具值。如图所示：

对于singer右手边的玩具，我们只要正常计数即可。但是它左手边的该如何计算呢？其实很简单，我们可以利用周期！。

Singer左手边的第3个玩具 = (singer的位置(1) + 周期(7) - 左手序数(3)) % 周期(7) = 5

即，Singer左手边的第3个玩具是正向第5号archer，从图上观察也是正确的。

看到这里，有没有同学突发奇想，那么正方向的是不是也可以这样数呢？答案是肯定的。

Singer右手边的第3个玩具 = (singer的位置(1) + 周期(7) + 右手序数(3)) % 周期(7) = 4

正好是顺序第4个玩具thinker。

如果singer是面朝外侧呢？很明显，我们只需要把上面两个式子进行交换即可。

在本题中玩具向内为0，向外为1。左数为0，右数为1。也就是说两个值同为0，或者同为1的时候，我们需要减掉序数，其余情况加上序数。

最终公式为：

玩具朝内计算左手边的第n个玩具 = 玩具朝外计算右手边的第n个玩具 = (玩具的位置 + 周期 - 左手序数) % 周期

其余则为加法。

**编码**

```
#include <bits/stdc++.h>

using namespace std;

#define MAX_N 100000

struct Toy {
    int dir;
    string occ;
};
//玩具信息数组
Toy toys[MAX_N];
//n个玩具，m组询问
int n, m;

int main() {
    //读入玩具的基础信息
   scanf("%d%d", &n, &m);
    for (int i = 0; i < n; ++i) {
        cin >> toys[i].dir >> toys[i].occ;
    }
    int cur = 0;
```

```
    //开始执行m次查询
    for (int i = 0; i < m; ++i) {
        int dir, s;
        // 获取左右和序数
        scanf("%d%d", &dir, &s);
        // 向外朝左与向内朝右都需要向反方向进行查找
        //即我当前的dir与搜索的dir相同时，都需要逆向查找。
        if (toys[cur].dir == dir) {
            s *= -1;
        }
        //负索引索对应的正数索引 = (当前索引+周期 + 负索引) % 周期
        cur = (cur + n + s) % n;
    }
    cout << toys[cur].occ;
    return 0;
}
```