# 第十九届全国信息学奥林匹克竞赛

# NOI 2002

## 第一试

| 题目名称 | 银河英雄传说 | 调皮的小孩 | 贪吃的九头龙 |
|---|---|---|---|
| 目录 | day1/galaxy | day1/child | day1/dragon |
| 可执行文件名 | galaxy | child | dragon |
| 输入文件名 | galaxy.in | | dragon.in |
| 输出文件名 | galaxy.out | | dragon.out |
| 是否有部分分 | 否 | 是 | 否 |
| 附加文件 | 无 | check | 无 |
| 时限 | 2 秒 | 5 秒 | 2 秒 |

注：每题 10 个测试点，共 100 分。

**竞赛时间：2002 年 8 月 12 日上午 8:00-13:00**

# 银河英雄传说

【问题描述】

公元五八〇一年，地球居民迁移至金牛座α第二行星，在那里发表银河联邦创立宣言，同年改元为宇宙历元年，并开始向银河系深处拓展。

宇宙历七九九年，银河系的两大军事集团在*巴米利恩星域*爆发战争。泰山压顶集团派宇宙舰队司令*莱因哈特*率领十万余艘战舰出征，气吞山河集团点名将*杨威利*组织麾下三万艘战舰迎敌。

杨威利擅长排兵布阵，巧妙运用各种战术屡次以少胜多，难免恣生骄气。在这次决战中，他将巴米利恩星域战场划分成 30000 列，每列依次编号为 1, 2, …, 30000。之后，他把自己的战舰也依次编号为 1, 2, …, 30000，让第 i 号战舰处于第 i 列(i = 1, 2, …, 30000)，形成“一字长蛇阵”，诱敌深入。这是初始阵形。当进犯之敌到达时，杨威利会多次发布合并指令，将大部分战舰集中在某几列上，实施密集攻击。合并指令为 M i j，含义为让第 i 号战舰所在的整个战舰队列，作为一个整体（头在前尾在后）接至第 j 号战舰所在的战舰队列的尾部。显然战舰队列是由处于同一列的一个或多个战舰组成的。合并指令的执行结果会使队列增大。

然而，老谋深算的莱因哈特早已在战略上取得了主动。在交战中，他可以通过庞大的情报网络随时监听杨威利的舰队调动指令。

在杨威利发布指令调动舰队的同时，莱因哈特为了及时了解当前杨威利的战舰分布情况，也会发出一些询问指令：C i j。该指令意思是，*询问电脑，杨威利的第 i 号战舰与第 j 号战舰当前是否在同一列中，如果在同一列中，那么它们之间布置有多少战舰。*

作为一个资深的高级程序设计员，你被要求编写程序分析杨威利的指令，以及回答莱因哈特的询问。

最终的决战已经展开，银河的历史又翻过了一页……

【输入文件】

输入文件 galaxy.in 的第一行有一个整数 T（1<=T<=500,000），表示总共有 T 条指令。

以下有 T 行，每行有一条指令。指令有两种格式：

1. M i j ：i 和 j 是两个整数（1<=i , j<=30000），表示指令涉及的战舰编号。该指令是莱因哈特窃听到的杨威利发布的舰队调动指令，并且保证第 i 号战舰与第 j 号战舰不在同一列。
2. C i j ：i 和 j 是两个整数（1<=i , j<=30000），表示指令涉及的战舰编号。该指令是莱因哈特发布的询问指令。

【输出文件】

输出文件为 galaxy.out。你的程序应当依次对输入的每一条指令进行分析和处理：

如果是杨威利发布的舰队调动指令，则表示舰队排列发生了变化，你的程序要注意到这一点，但是不要输出任何信息；

如果是莱因哈特发布的询问指令，你的程序要输出一行，仅包含一个整数，

表示在同一列上，第 i 号战舰与第 j 号战舰之间布置的战舰数目。如果第 i 号战舰与第 j 号战舰当前不在同一列上，则输出-1。

【样例输入】
4
M 2 3
C 1 2
M 2 4
C 4 2

【样例输出】
-1
1

【样例说明】
战舰位置图：表格中阿拉伯数字表示战舰编号

| | 第一列 | 第二列 | 第三列 | 第四列 | …… |
|---|---|---|---|---|---|
| 初始时 | 1 | 2 | 3 | 4 | …… |
| M 2 3 | 1 | | 3<br>2 | 4 | …… |
| C 1 2 | 1 号战舰与 2 号战舰不在同一列，因此输出-1 | | | | |
| M 2 4 | 1 | | | 4<br>3<br>2 | …… |
| C 4 2 | 4 号战舰与 2 号战舰之间仅布置了一艘战舰，编号为 3，输出 1 | | | | |

# 调皮的小孩

【问题描述】

一群小孩在草坪上玩游戏，十分开心，一个喜欢猎奇的过路人走过来问他们：

“孩子们，你们在玩什么游戏呢？”

“我们中有一个人当裁判，剩下的人分成两队：星星队有 N 个人，月亮队有 M 个人。如果你猜对了谁是裁判，我就告诉你玩的是什么游戏。”

“好啊。不过，总得给我点提示吧？”

“那当然。你可以问我们某人是不是属于某队，而不能问某人是不是裁判。被问到的星星队的队员总是告诉你正确的答案；月亮队的队员总是告诉你错误的答案；而裁判，在你向他问奇数次的时候他会告诉你正确的答案，偶数次的时候会告诉你错误的答案。”

“哦，明白了。可以随便提问题吗？”

“你不许问任何人关于他自己的问题。例如，你不许问我：‘你是不是星星队的？’你也不能向任何一个人询问两次关于同一个人的问题。例如，你曾问过我丁丁是不是星星队的，你就不能再问我丁丁是不是月亮队的。最后，*请你尽量不要问同一个人太多的问题*，因为他还要接着玩呢，没时间老回答你的问题。”

过路人很聪明，不仅猜出了谁是裁判，还说出了剩下的每个人是哪个队的。你也来试试吧！

【交互】

本题是一道交互式题目，*你的程序应当和测试库进行交互，而不得访问任何文件。*测试库提供三个函数：GetNM，Ask，Answer，它们的作用和用法如下：

- GetNM(N,M)必须首先调用，用它来获得正整数 N,M 的值。(2<=N+M<=500)。
- Ask(Child1,Child2,T)的作用是询问。其中 1<=Child1,Child2<=N+M+1,且 Child1≠Child2。T 非 0 即 1，T 为 0 表示星星队，为 1 表示月亮队。即询问小孩 Child1“小孩 Child2 是不是属于 T 队”。若函数返回 1，表示 Child1 回答说“是”；若函数返回 0，表示 Child1 回答“否”。
- Answer(Ans)用来告诉测试库你猜的答案。参数 Ans 的值为 0，1，2。为 0 表示星星队，为 1 表示月亮队，为 2 表示裁判。你应当连续调用 N+M+1 次本过程，从 1 号开始到 N+M+1 号为止依次说明每个小孩的角色，注意仅有一个裁判。调用完 N+M+1 次本过程后，测试库会终止你的程序，切记*你的程序不得自行终止。*

【一个成功交互的例子】

| 函数调用 | 返回值 | 说明 |
|---|---|---|
| GetNM(N,M) | N=1, M=1 | 星星队和月亮队各有一名队员 |
| Ask(1,2,0) | 0 | 问小孩 1：“小孩 2 是不是星星队的？”答：“否” |
| Ask(2,1,0) | 1 | 问小孩 2：“小孩 1 是不是星星队的？”答：“是” |
| Ask(3,1,1) | 0 | 问小孩 3：“小孩 1 是不是月亮队的？”答：“否” |
| Answer(2) | 无 | 小孩 1 是裁判。 |
| Answer(1) | 无 | 小孩 2 是月亮队的。 |
| Answer(0) | 无 | 小孩 3 是星星队的。 |

【对 Pascal 程序员的提示】
你的程序应当使用下列语句引用测试库：
uses childlib;
测试库提供的函数/过程原型为：
procedure GetNM(var N,M:integer);
function Ask(Child1,Child2,T:integer):integer;
procedure Answer(Ans:integer);

【对 C/C++程序员的提示】
你应当建立一个工程，把文件 childlib.o 包含进来，然后在程序头加上一行：
#include “childlib.h”
测试库提供的函数原型为：
void GetNM(int *N, int *M);
int Ask(int Child1, int Child2, int T);
void Answer(int Ans);

【评分方法】
如果你的程序有下列情况之一，得 0 分：

- 访问了任何文件(包括临时文件)或者自行终止；
- 非法调用库函数；
- 让测试库异常退出。

否则每个测试点你的得分按这样来计算：

1. *你只猜对了裁判是谁而没有完全猜对其余孩子所在的队。*在这种情况下，如果你对某个小孩提了三个以上（含三个）的问题，那么你只能得 40%的分，否则可以得 60%的分；
2. *你猜对了裁判是谁以及其余所有孩子所在的队。*在这种情况下，如果你对某个小孩提了三个以上（含三个）的问题，那么你只能得 70%的分，否则你将得到该测试点的满分。

【你如何测试自己的程序】

1. 在工作目录下建立一个文本文件 child.in，文件第一行包括两个整数 N，M，第二行包括 N+M+1 个数（数的取值为 0,1,2），第 k 个数为小孩 k 所在的队，0 表示星星队，1 表示月亮队，2 表示裁判。样例输入文件存放在用户目录中。
2. 执行你的程序，此时测试库会产生输出文件 child.log。
3. 如果程序正常结束，child.log 的第一行包含一个整数 P，即被询问次数最多的小孩被问了多少次（超过 10 次的按 10 次计）。第二行包含 N+M+1 个数，依次为你的程序对每个孩子的猜测结果。
   如果程序非法退出，则 child.log 会记录如下内容:“Abnormal Termination”。
4. 在工作目录下执行程序 check，会在屏幕上看到你的得分。

# 贪吃的九头龙

【问题描述】

传说中的九头龙是一种特别贪吃的动物。虽然名字叫“九头龙”，但这只是说它出生的时候有九个头，而在成长的过程中，它有时会长出很多的新头，头的总数会远大于九，当然也会有旧头因衰老而自己脱落。

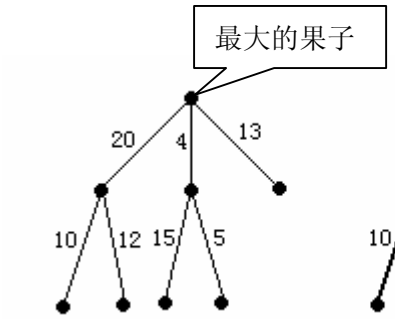
有一天，有 M 个脑袋的九头龙看到一棵长有 N 个果子的果树，喜出望外，恨不得一口把它全部吃掉。可是必须照顾到每个头，因此它需要把 N 个果子分成 M 组，每组至少有一个果子，让每个头吃一组。

这 M 个脑袋中有一个最大，称为“大头”，是众头之首，*它要吃掉恰好 K 个果子，而且 K 个果子中理所当然地应该包括唯一的一个最大的果子。*果子由 N-1 根树枝连接起来，由于果树是一个整体，因此可以从任意一个果子出发沿着树枝“走到”任何一个其他的果子。

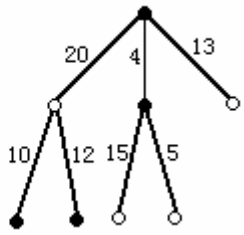
对于每段树枝，如果它所连接的两个果子需要由不同的头来吃掉，那么两个头会共同把树枝弄断而把果子分开；如果这两个果子是由同一个头来吃掉，那么这个头会懒得把它弄断而直接把果子连同树枝一起吃掉。当然，吃树枝并不是很舒服的，因此每段树枝都有一个吃下去的“难受值”，而九头龙的难受值就是所有头吃掉的树枝的“难受值”之和。

九头龙希望它的“难受值”尽量小，你能帮它算算吗？

例如图 1 所示的例子中，果树包含 8 个果子，7 段树枝，各段树枝的“难受值”标记在了树枝的旁边。九头龙有两个脑袋，大头需要吃掉 4 个果子，其中必须包含最大的果子。即 N=8，M=2，K=4：




图一




大头吃 4 个果子，用实心点标识；
小头吃 4 个果子，用空心点标识；
九头龙的难受值为 4，因为图中用细边标记的树枝被大头吃掉了。

图二

图一描述了果树的形态，图二描述了最优策略。

【输入文件】

输入文件 dragon.in 的第 1 行包含三个整数 N (1<=N<=300)，M (2<=M<=N)，K (1<=K<=N)。 N 个果子依次编号 1,2,...,N，且*最大的果子的编号总是 1*。第 2 行到第 N 行描述了果树的形态，每行包含三个整数 a (1<=a<=N)，b (1<=b<=N)，c (0<=c<=$10^5$)，表示存在一段难受值为 c 的树枝连接果子 a 和果子 b。

【输出文件】

输出文件 dragon.out 仅有一行，包含一个整数，表示在满足“大头”的要求的前提下，九头龙的难受值的最小值。如果无法满足要求，输出-1。

【样例输入】

8 2 4
1 2 20
1 3 4
1 4 13
2 5 10
2 6 12
3 7 15
3 8 5

【样例输出】

4

【样例说明】

该样例对应于题目描述中的例子。