第二十届全国青少年信息学奥林匹克竞赛 NOI2003

第一试

题目名称	木棒游戏	文本编辑器	卫星探测
目录	day1/game	day1/editor	day1/detect
题目类型	普通	普通	交互
可执行文件名	game	editor	detect
输入文件名	game.in	editor.in	-
输出文件名	game.out	editor.out	-
是否有部分分	否	否	是
题目总分	100	100	100
时间限制	1 秒	2~4 秒	1 秒
内存限制	64M	64M	64M

有关附加文件的信息，请参看具体的题目说明。

木棒游戏

【问题描述】

这是一个很古老的游戏。用木棒在桌上拼出一个**不成立的等式**，移动且只移动一根木棒使得等式成立。现在轮到你了。

【任务】

从文件读入一个式子（该式子肯定是一个不成立的等式）。

如果移动一根木棒可以使等式成立，则输出新的等式，否则输出 No。

【说明和限制】

1．式子中的数可能是正数或负数，运算符号只会出现加号和减号，并且有且仅有一个等号，不会出现括号、乘号或除号，也不会有++,--,+-或-+出现。

2．式子中不会出现 8 个或 8 个以上的连续数字（数的绝对值小于等于 9999999）。

3．你只能移动用来构成数字的木棒，不能移动构成运算符（+、-、=）的木棒，所以加号、减号、等号是不会改变的。移动前后，木棒构成的数字必须严格与图 2 中的 0~9 相符。

4．从文件读入的式子中的数不会以 0 开头，但允许修改后等式中的数以数字 0 开头。

【输入数据】

从文件 game.in 中读入一行字符串。该串中包括一个以“#”字符结尾的式子（ASCII 码 35)，式子中没有空格或其他分隔符。输入数据严格符合逻辑。字符串的长度小于等于 1000。

注意：“#”字符后面可能会有一些与题目无关的字符。

【输出数据】

将输出结果存入文件 game.out，输出仅一行。

如果有解，则输出正确的等式，格式与输入的格式相同（以“#”结尾，中间不能有分隔符，也不要加入多余字符）。此时输入数据保证解是唯一的。

如果无解，则输出“No”（N 大写，o 小写）。

【输入样例 1】

1+1=3#

【输出样例 1】

1+1=2#

【输入样例 2】

1+1=3+5#

【输出样例 2】

No

【输入样例 3】

11+77=34#

【输出样例 3】

17+17=34#

文本编辑器

【问题描述】

很久很久以前，DOS3.x 的程序员们开始对 EDLIN 感到厌倦。于是，人们开始纷纷改用自己写的文本编辑器……

多年之后，出于偶然的机会，小明找到了当时的一个编辑软件。进行了一些简单的测试后，小明惊奇地发现：那个软件每秒能够进行上万次编辑操作（当然，你不能手工进行这样的测试）！于是，小明废寝忘食地想做一个同样的东西出来。你能帮助他吗？

为了明确任务目标，小明对“文本编辑器”做了一个抽象的定义：

文本：由 0 个或多个字符构成的序列。这些字符的 ASCII 码在闭区间[32, 126]内，也就是说，这些字符均为可见字符或空格。

光标：在一段文本中用于指示位置的标记，可以位于文本的第一个字符之前，文本的最后一个字符之后或文本的某两个相邻字符之间。

文本编辑器：为一个可以对一段文本和该文本中的一个光标进行如下六条操作的程序。如果这段文本为空，我们就说这个文本编辑器是空的。

操作名称	输入文件中的格式	功能
MOVE(k)	Move k	将光标移动到第 k 个字符之后，如果 k=0，将光标移到文本第一个字符之前
INSERT(n, s)	Insert n↵ S	在光标后插入长度为 n 的字符串 s，光标位置不变，$n \geq 1$
DELETE(n)	Delete n	删除光标后的 n 个字符，光标位置不变，$n \geq 1$
GET(n)	Get n	输出光标后的 n 个字符，光标位置不变，$n \geq 1$
PREV()	Prev	光标前移一个字符
NEXT()	Next	光标后移一个字符

比如从一个空的文本编辑器开始，依次执行操作 `INSERT(13, “Balanced□tree”)`，`MOVE(2)`，`DELETE(5)`，`NEXT()`，`INSERT(7, “□editor”)`，`MOVE(0)`，`GET(15)`后，会输出“`Bad□editor□tree`”，如下表所示：

操作	操作前的文本	操作后的结果
INSERT(13, “Balanced□tree”)	**|** （只有光标，文本为空）	**|**`Balanced□tree`
MOVE(2)	**|**Balanced□tree	`Ba`**|**`lanced□tree`
DELETE(5)	Ba**|**lanced□tree	`Ba`**|**`d□tree`
NEXT()	Ba**|**d□tree	`Bad`**|**`□tree`
INSERT(7,	Bad**|**□tree	Bad**|**□editor□tree

“□editor”)，		
MOVE(0)	Bad|□editor□tree	|Bad□editor□tree
GET(15)	|Bad□editor□tree	输出“Bad□editor□tree”

上表中，“|”表示光标，“□”表示空格。

你的任务是：

- 建立一个空的文本编辑器。
- 从输入文件中读入一些操作指令并执行。
- 对所有执行过的 GET 操作，将指定的内容写入输出文件。

【输入文件】

输入文件 editor.in 的第一行是指令条数 *t*，以下是需要执行的 *t* 个操作。其中：

为了使输入文件便于阅读，Insert 操作的字符串中可能会插入一些回车符，请忽略掉它们（如果难以理解这句话，可以参考样例）。

除了回车符之外，输入文件的所有字符的 ASCII 码都在闭区间[32, 126]内。且行尾没有空格。

这里我们有如下假定：

- MOVE 操作不超过 50000 个，INSERT 和 DELETE 操作的总个数不超过 4000，PREV 和 NEXT 操作的总个数不超过 200000。
- 所有 INSERT 插入的字符数之和不超过 2M（1M=1024*1024），正确的输出文件长度不超过 3M 字节。
- DELETE 操作和 GET 操作执行时光标后必然有足够的字符。MOVE、PREV、NEXT 操作不会把光标移动到非法位置。
- 输入文件没有错误。

对 C++选手的提示：经测试，对最大的测试数据使用 fstream 进行输入有可能会比使用 stdio 慢约 1 秒，因此建议在可以的情况下使用后者。

【输出文件】

输出文件 editor.out 的每行依次对应输入文件中每条 GET 指令的输出。

【样例输入】

```
15
Insert 26
abcdefghijklmnop
qrstuv wxy
Move 15
Delete 11
Move 5
Insert 1
^
Next
Insert 1
```

_
Next
Next
Insert 4
.\/.
Get 4
Prev
Insert 1
^
Move 0
Get 22

【样例输出】

.\/.
abcde^_^f.\/.ghijklmno

卫星探测

【问题描述】

A 国最近检测到了 B 国内有不正常的辐射，经调查发现，这是因为 B 国正在耗资百亿研制新式武器——连环阵。可是，由于 B 国对此武器的高度保密措施，A 国的间谍甚至无法确定出连环阵的具体位置。不过，A 国的卫星还是可以找出连环阵所在的基地的。我们现在知道该基地是一个边上含有放射性物质的凸多边形。经研究发现，在这个凸多边形所在的平面内，它具有如下性质：

- 包含坐标原点；
- 任意两条边不在同一直线上；
- 没有和 *x* 轴或 *y* 轴平行的边；
- 所有顶点的 *x* 坐标和 *y* 坐标都是整数。

现在 A 国可以通过卫星发出无限大的扇形探测波，与该凸多边形所在平面交于一条直线。不过该直线不是与 *x* 轴平行，就是与 *y* 轴平行。通过反射信号，我们可以确定该直线与凸多边形的交点个数和所有交点的坐标（如果有的话）。

现在，需要你写一个程序，通过控制卫星发出的探测波来确定这个凸多边形。

【交互方法】

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 3 组函数，分别用于程序的初始化，与测试库的交互，以及返回结果。它们的用法与作用如下：

- `prog_initialize` **必须先调用**，但只能调用一次，用作初始化测试库；
- 测试库提供两个函数 `ask_x` 和 `ask_y` 作为与测试库交互的方式。其中 `ask_x(`x_0`, `y_1`, `y_2`)`的的作用是询问直线 $x=x_0$ 和多边形的交点，`ask_y(y0, x1, x2)`的作用是询问直线 $y=y_0$ 与多边形的交点，函数的返回值是交点的个数。`ask_x(`x_0`, `y_1`, `y_2`)`调用后，y_1 和 y_2 被赋值为交点的 y 坐标；`ask_y(`y_0`, `x_1`, `x_2`)`调用后，x_1 和 x_2 被赋值为交点的 x 坐标。如果只有一个交点，那么 x_1 和 x_2 或 y_1 和 y_2 的值相同；如果没有交点，那么 x_1 和 x_2 或 y_1 和 y_2 的值没有意义。
- 最后的一组函数是你的程序用来向测试库返回结果的。这里包括返回多边形面积的 `ret_area(s)`，返回多边形顶点数目的 `ret_n(n)`，返回多边形顶点坐标的 `ret_vertex(x, y)`。需要注意的事，这里 `ret_area` 是必须先于 `ret_n` 调用的，而 `ret_n` 又是必须先于 `ret_vertex` 调用的。不合适的调用方式将会强制你的程序非法退出。这里你需要在调用 `ret_n` 后调用 n 次 `ret_vertex` 返回多边形的顶点，但需要注意的是，如果你用 `ret_n` 返回的结果是错误的，那么测试库将会马上终止你的程序，并不接受下面的结果；同样的，如果 `ret_vertex` 中返回了错误的结果，那么测试库也会马上终止你的程序。如果 `ret_vertex` 的结果均是正确的，那么测试库将会在你返回最后一个顶点坐标后终止你的程序。
- **注意：你需要按照逆时针顺序返回所有顶点的坐标，不过你可以从任意一个顶点开始。**

【对使用 Pascal 选手的提示】

你的程序应当使用如下的语句引用测试库。

```
uses detect_lib;
```

测试库使用的函数原型为：

```
procedure prog_initialize;
function ask_x(const x0: longint; var y1, y2: double): longint;
function ask_y(const y0: longint; var x1, x2: double): longint;
procedure ret_area(const s: double);
procedure ret_n(const n: longint);
procedure ret_vertex(const x, y:longint);
```

【对使用 C/C++选手的提示】

你应当建立一个工程，把文件 libdetect.o 包含进来，然后在程序头加上一行：

```
#include “detect_lib.h”
```

测试库使用的函数原型为：

```
void prog_initialize();
int ask_x(int x0, double *y1, double *y2);
int ask_y(int y0, double *x1, double *x2);
void ret_area(double s);
void ret_n(int n);
void ret_vertex(int x, int y);
```

【数据说明】

如果凸多边形的坐标如右图所示，那么一种可能得满分的调用方案如下：

Pascal 选手的调用方法	C/C++选手的调用方法	说明
Prog_initialize;	prog_initialize();	初始化程序
ask_x(-6, y1, y2);	ask_x(-6, &y1, &y2);	返回值 1，$y_1=2$，$y_2=2$
ask_x(-5, y1, y2);	ask_x(-5, &y1, &y2);	返回值 2，$y_1=3.4$，$y_2=-5$
ask_y(2, x1, x2);	ask_y(2, &x1, &x2);	返回值 2，$x_1=-6$，$x_2=16$
ask_y(-20, x1, x2)	ask_y(-20, &x1, &x2)	返回值 0，x_1、x_2中的值无意义
ret_area(241.5);	ret_area(241.5);	返回面积
ret_n(5);	ret_n(5);	返回 n
ret_vertex(8, -9);	ret_vertex(8, -9);	返回顶点(8, -9)
ret_vertex(16, 2);	ret_vertex(16, 2);	返回顶点(16, 2)
ret_vertex(-1, 9);	ret_vertex(-1, 9);	返回顶点(-1, 9)
ret_vertex(-6, 2);	ret_vertex(-6, 2);	返回顶点(-6, 2)
ret_vertex(-5, -5);	ret_vertex(-5, -5);	返回顶点(-5, -5)

注意，该例子只是对库函数的使用说明，并没有算法上的意义。

这里 n 最大为 200，x、y 坐标在[-10000, 10000]这个区间内。

【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：包括顶点数提交正确的 1 分，面积提交正确的 2 分，顶点坐标完全正确的 2 分，分数累计。剩下的 5 分将根据你调用 ask_x 和 ask_y 的总次数进行评判，公式如下：

这里 *x* 为你的程序调用的 ask_x 和 ask_y 的次数，*score* 为你的得分。

【你如何测试自己的程序】

1. 在工作目录下建立一个文件叫做 detect.in，文件的第一行包括一个整数 *n* 为顶点的数目，以下 *n* 行每行两个整数**按照逆时针方向**给出凸多边形的顶点坐标；
2. 执行你的程序，此时测试库会产生输出文件 detect.log，该文件中包括了你程序和库交互的记录和最后的结果；
3. 如果程序正常结束，detect.log 的最后一行会给出你的程序的分数；
4. 如果程序非法退出，则我们不保证 detect.log 中的内容有意义。