Programming Logo

Filename: logo

Logo is an old programming language with a turtle that holds a pen. The commands in the language allow the turtle to move in the direction he is positioned. If the pen is down, then a line is drawn in his path of movement. The UCF Programming Team has decided that it wants to resurrect Logo and market the new tool to middle schools in the hope of inspiring the next wave of UCF Programming Team Champions.

In NewLogo, the turtle draws on a grid, 20 characters high and 30 characters wide with the top left corner with the coordinates (0,0) and the bottom right corner with the coordinates (19, 29). Each grid square will contain a single text character. At the beginning of any NewLogo program, the grid contains all blank squares and the turtle is located at (0,0) headed right with his pen down. In addition to being able to put his pen up and down, the turtle can change the character with which his pen draws (his starting character is '*'). This will allow the display to have more variety than before. Finally, the turtle can turn right or left in increments of 45 degrees, so it is possible to move diagonally. Here are the NewLogo commands:

START
PEN UP
PEN DOWN
CHANGE CHARACTER C
MOVE X
RIGHT TURN R
LEFT TURN L
END

The START command simply indicates the start of a NewLogo program while the END command indicates the end of a NewLogo program. Both PEN commands do not move the turtle at all, but simply change the state of the turtle's pen to either UP (not writing) or DOWN (writing) as indicated. The CHANGE CHARACTER command does not move the turtle at all or draw anything. It simply changes the character with which the pen will draw (if it is down) on subsequent commands after this command is executed.

The MOVE command simply moves the turtle from its current spot X number of steps. If the pen is down, then this character is drawn in each grid square on the path. The only square for which this isn’t necessarily true is the initial square of the path. If this initial square was previously drawn on, then we don’t draw over it with the potentially new character. If this initial square was previously blank, then we do draw over the square with the new character. Thus, if the first line after START in a program is MOVE 5, then 6 stars are drawn. However, if this is followed by MOVE 3, only 3 more stars will be drawn. The RIGHT TURN and LEFT TURN commands turns the turtle, in place, R degrees to the right or L degrees to the left, respectively (R and L are given in multiples of 45 degrees).

If a NewLogo program commands the turtle to go off the 20 x 30 grid, the program has caused a TURTLE OUT OF BOUNDS ERROR.
The Problem:

For each NewLogo Program, produce the output of the program. You are guaranteed that these programs are syntactically correct. Namely, only the commands listed above will be used, the angles for the turns will be positive integers that are multiples of 45, the values for movement will be positive integers and the character for the change character command will always be a non-whitespace printable character (these are characters with ASCII values in between 33 and 126, inclusive). If the program produces a TURTLE OUT OF BOUNDS ERROR, then simply display this error message instead.

The Input:

The first line of the input file will contain a single positive integer, $n$ ($n < 100$), designating the number of NewLogo programs your program will have to interpret. All of the programs follow, with one command per line. You are guaranteed that the first line of each program is the command “START” and that the last line of each program is the command “END”. On each line of code, all words and numbers will be separated by a single space from each other. No extra spaces will be included at the beginning or end of any lines.

The Output:

The first line of output for each program should have the following format:

```
Program #k
```

where $k$ represents the program number, starting at 1.

If there is no error in the program, then output the entire grid. In order for the output of a program to be accurate, there must be spaces (not tabs or other characters) in each spot where no output was written during the course of the program. In addition, place the grid inside of a rectangle of +, - and | characters where - makes up the horizontal line, | makes up the vertical line and + represents the corners (see Sample Output). Each grid will be output as 22 lines of 32 characters each (which is the 20 by 30 grid surrounded by the rectangle).

If there is a TURTLE OUT OF BOUNDS error in the program, simply identify the line number of the first time the error occurs and output a single line with the following format:

```
LINE x: TURTLE OUT OF BOUNDS ERROR
```

where $x$ represents the line number of the first line of code in the program that causes a TURTLE OUT OF BOUNDS ERROR. The START line is line number 0. All subsequent lines are numbered in regular counting order.

Skip a blank line in between the output for each program.
Sample Input:

2
START
MOVE 5
RIGHT TURN 90
CHANGE CHARACTER &
MOVE 4
RIGHT TURN 45
CHANGE CHARACTER @
MOVE 3
END
START
MOVE 25
RIGHT TURN 90
MOVE 5
LEFT TURN 90
MOVE 6
RIGHT TURN 45
END

Sample Output:

Program #1

```
******
 &
 &
 &
 &
 @
 @
```

Program #2

LINE 5: TURTLE OUT OF BOUNDS ERROR