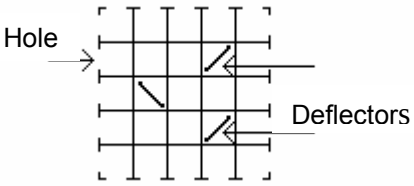
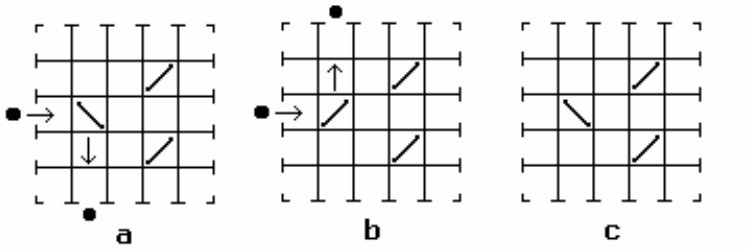
A BLACK BOX GAME

The Black Box Game is played with a square-shaped black box lying flat on a table. Each of its four sides has *n* holes (for a total of 4*n* holes) into which a ball can be thrown. A thrown ball will eventually exit from one of the 4*n* holes, potentially the same hole into which it was thrown.

The black box's internals can be envisioned as an *n* x *n* grid. The holes in the sides are the starts and ends of rows and columns. Each of the box's squares is either empty or occupied by a *deflector*. A deflector is a piece of hardware that changes the direction of the ball by 90 degrees. Consider this example of a 5x5 box.



A ball thrown into the box follows a straight line until it either hits a deflector or exits the box. When a ball hits a deflector, the ball changes direction and the deflector toggles its position (by "toggle" we mean rotate 90 degrees). The examples below show the action of a deflector.



a) A ball is thrown through a hole; it hits a deflector and changes direction.
b) After the first ball was thrown, the deflector has toggled its position. A new ball is thrown into the same hole, hits the deflector and is deflected in a direction opposite to that of the first ball.
c) The deflector toggles every time it is hit.

Whenever a deflector is hit, it makes a beep. The number of times the ball was deflected can be deduced by counting the beeps. It can be proved that the ball always exits the box. The box has a button that resets it to its original state and another button that toggles all of its deflectors.

TASK

You will be provided with an interface to 15 black boxes via a library of Pascal or C/C++ functions. You must determine the internals of each one of them as best as possible and submit a file describing each. You will also be provided with a method to define your own black boxes for testing.

CONSTRAINTS

$1 \le n \le 30$

OUTPUT

You must submit a file containing the following data for each one of the 15 black-boxes

`blackboxK.out`	DESCRIPTION

`#FILE blackbox K`
`.....`
`.../.`
`.\...`
`.../.`
`.?.?.`
	LINE 1: The file header. The file header must contain
	`#FILE blackbox K`
	where K (range 1..15) corresponds to the box being solved.
	***n* LINES:** Each line describes a row of the box, starting from the topmost row to the bottom row. Each line must contain exactly *n* characters; each character corresponds to a column (running from left to right).
	• '.' means that the square is empty.
	• '/' means the square contains a deflector with initial position '/'
	• '\' means the square contains a deflector with initial position '\'
	• '?' means that you were unable to determine the initial contents of that square

LIBRARY

You are given a library that provides the following functions

FUNCTION	Description

PASCAL
`function Initialize(box: integer): integer;`
C/C++
`int Initialize(int box);`
	Initializes the library, must be called once at the start of your program. It returns *n*, the number of holes on each side of the box.
	The parameter *box* must contain an integer between 1 and 15 indicating the box you want to use or 0 if you want to use a box created by you.

PASCAL
`function throwBall(holeIn, sideIn: integer;`
`var holeOut, sideOut: integer): longint;`
C
`int throwBall(int holeIn, int sideIn,`
`int *holeOut, int *sideOut);`
C++
`int throwBall(int holeIn, int sideIn,`
`int &holeOut, int &sideOut);`
	Throws a ball into the box through hole number *holeIn* in side *sideIn*, sides are numbered as 1 – Top, 2 – Right, 3 – Bottom and 4 – Left. Holes are numbered from left to right and from top to bottom starting from number 1 in each side. In *holeOut* and *sideOut* you'll receive the hole and side number where the ball exits the box. The function *throwBall* returns the number of beeps caused by the ball hitting a deflector.

PASCAL
`procedure ResetBox;`
C/C++
`void ResetBox();`
	Resets every deflector in the box to its initial position.

PASCAL
`procedure ToggleDeflectors;`
C/C++
`void ToggleDeflectors();`
	Toggles every deflector in the box.

PASCAL
`procedure Finalize;`
C/C++
`void Finalize();`
	Gracefully ends the interaction with the box. It should be called at the end of your program.

To be able to use the library in your program do as follows:

- **FreePascal:** In the task directory you will find the files `pbblib.o` and `pbblib.ppu` to be able to use them include the following statement.
  ```
  uses pbblib;
  ```
  The file `pblackbox.pas` gives an example of how to use the library.

- **C:** In the task directory you will find the files `cbblib.o` and `cbblib.h`, to be able to use them include the following statement in your code.
  ```
  #include "cbblib.h"
  ```
  The file `cblackbox.c` gives an example of how to use the library. In order to compile your code you will need to use the following command
  ```
  gcc –o yourprogram cbblib.o yourprogram.c
  ```

- **C++:** In the task directory you will find the files `cppbblib.o` and `cppbblib.h`, to be able to use them include the following statement in your code.
  ```
  #include "cppbblib.h"
  ```
  The file `cppblackbox.cpp` gives an example of how to use the library. In order to compile your code you will need to use the following command
  ```
  g++ –o yourprogram cppbblib.o yourprogram.cpp
  ```

NOTE: At any given time only one program using the library can be running.

SAMPLE INTERACTION

A sample interaction for the box in the previous figure could be:

FUNCTION CALL	VALUE RETURNED BY FUNCTION
`Initialize(0);`	Assuming that the box used is the one in the previous figure. Returns 5 indicating that *n* = 5

PASCAL
`throwBall(3,4,holeOut,sideOut);`
C
`throwBall(3,4,&holeOut,&sideOut);`
C++
`throwBall(3,4,holeOut,sideOut);`
	A ball is thrown into hole number 3 (third from the top) on the left side. Returns 1, indicating that the ball hit 1 deflector. When the function returns, *holeOut* will equal 2 and *sideOut* will equal 3 indicating that the ball exited through hole 2 (second from the left) of the bottom side of the box.

EXPERIMENTATION

If you pass an integer value of 0 to the function `Initialize`, the library will read the box internals from the file `blackbox.in`. In this way you can experiment with the library. The format for the file `blackbox.in` is described below.

`blackbox.in`	DESCRIPTION

`5`
`3`
`2 3 \`
`4 2 /`
`4 4 /`
	LINE 1: Contains *n,* the number of holes on each side.
	LINE 2: Contains an integer *d* that indicates the number of deflectors in the box.
	NEXT *d* LINES: There must be one line for every deflector in the box. Each line contains two space-separated integers that represent the column and the row of the deflector respectively, and a character separated with a space from the second integer that can be either '/' or '\' that represents the original position of the deflector.

NOTE; The example blackbox.in describes the black box shown in the figure at the top of page 1.

ERROR MESSAGES

In the case of any anomalies the library will output an error message to the standard error. The possible error message that you can get and their meanings are shown in the table below.

ERROR MESSAGE	MEANING
ERR 1 More than one app	Only one application at a time can interact with the black boxes, please restart all applications and start only one at a time
ERR 2 Invalid box	The box number you input is not in the range from 0 to 15
ERR 3 Invalid deflector	The file blackbox.in has a deflector in an invalid position
ERR 4 Invalid symbol	The file blackbox.in has an invalid symbol
ERR 5 Invalid size	The size of the black box in blackbox.in is invalid.
ERR 6 Invalid input hole	Either the side or the input hole that you entered are invalid
ERR 7 ALARM	Please call the technical staff

GRADING

For each box you must submit a text file that describes the internals of the black box as best as possible. For each box:

- If your submission has a '.', '/' or '\' character in an incorrect position, you'll get zero points for that box.
- Let B_m be the maximum number of discovered positions among all of the correct submissions, and let B_y be the number of discovered positions in your submission, then percentage of your score for that box will be

$$100\ B_y / B_m$$

NOTE: The official solution for this task can programmatically discover 100% of the initial contents of any of the boxes in a time less than 8 minutes.