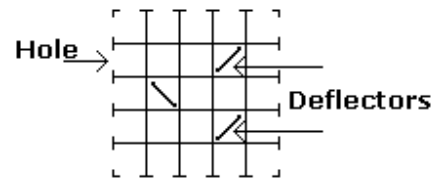




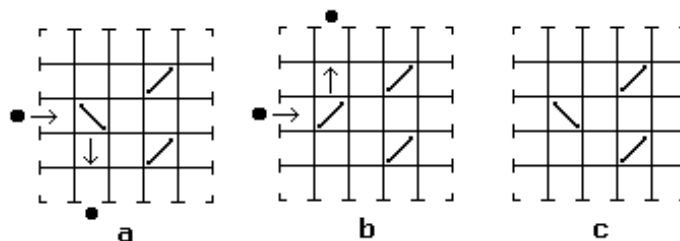
**BLACK BOX, HET SPEL**

Het spel Black Box wordt gespeeld met een vierkante zwarte doos die plat op tafel ligt. In alle vier zijden zitten  $n$  gaten (in totaal dus  $4n$  gaten) waar je een bal in kunt schieten. Een bal die je erin schiet komt uiteindelijk weer door een van de  $4n$  gaten naar buiten toe; dit kan ook het gat zijn waardoor je de bal naar binnengeschooten hebt.

De binnenkant van de doos kun je je voorstellen als een grid van  $n \times n$ . De gaten in de zijanten van de doos zijn telkens het begin en einde van de rijen en kolommen. De vierkantjes in de binnenkant zijn ofwel leeg, of er zit een kaatser (*deflector*) in. Een kaatser is een apparaatje dat de richting van de bal 90 graden aanpast. Hier zie je een voorbeeld van een  $5 \times 5$  doos.



Als je een bal in de doos schiet dan volgt deze een rechte lijn totdat hij ofwel een kaatser raakt, of de doos weer verlaat. Als een bal een kaatser raakt dan verandert de bal van richting en wordt de stand van de kaatser aangepast (dat betekent 90 graden gedraaid). Het volgende voorbeeld laat je zien hoe een kaatser werkt.



- a) Een bal wordt in het gat geschoten; de bal raakt een kaatser en verandert van richting
- b) Nadat de eerste bal geschoten is verandert de kaatser van stand. De volgende bal die de kaatser raakt wordt nu dus de andere kant op van richting veranderd.
- c) De stand van de kaatser wordt iedere keer als hij geraakt wordt aangepast.

Als een kaatser wordt geraakt geeft hij een piepje. Het aantal keren dat kaatsers worden geraakt kun je afleiden uit het aantal piepjes. Je kunt bewijzen dat de bal altijd weer naar buiten komt. De doos heeft een knop om alle kaatsers weer in de beginpositie te zetten en een andere knop waarmee alle kaatsers in hun andere positie kunnen worden gezet.

**OPDRACHT**

Er zijn 15 beschrijvingen van dozen beschikbaar via een Pascal of C/C++ library. Je moet zo goed mogelijk bepalen hoe deze dozen er van binnen uit zien en vervolgens een bestand inzenden met een beschrijving van de binnenkant ervan. Je krijgt ook de mogelijkheid om je eigen black boxes te maken om mee te testen.

**RANDVOORWAARDE**

$$1 \leq n \leq 30$$

**UITVOER**

Je moet voor elke doos een bestand opleveren met een beschrijving van de binnenkant.

blackboxK.out	BESCHRIJVING
#FILE blackbox K ..... .../. .\... .../. .?.?.	Schrijf naar de eerste regel de header van het bestand. De header ziet er als volgt uit: #FILE blackbox K Schrijf naar de volgende $n$ regels telkens een regel die een rij van de doos beschrijft. Begin met de bovenste rij. Op elke regel moet je exact $n$ karakters schrijven. Elk karakter komt overeen met een vierkant in een kolom. De volgende karakters worden gebruikt:



	<p>'.' betekent dat het vierkant leeg is.</p> <p>'/' betekent dat er een kaatser in het vierkant staat met initiële stand '/'.          '\ ' betekent dat er een kaatser in het vierkant staat met initiële stand '\'.          '?' betekent dat je niet kon bepalen wat de initiële inhoud van het vierkant was.</p>
--	---

**LIBRARY**

Je krijgt een library met de volgende functies

FUNCTIE	Beschrijving
<p><b>PASCAL</b>  <code>function Initialize(box: integer): integer;</code></p> <p><b>C/C++</b>  <code>int Initialize(int box);</code></p>	<p>Deze functie initialiseert de library. Deze functie moet je eenmaal aanroepen als je programma gestart wordt. Het resultaat van de functie is het <math>n</math>, aantal gaten in elke zijde van de doos.</p> <p>De parameter <math>box</math> (<math>0 \leq box \leq 15</math>) geeft aan met welke doos je wilt spelen. Als de parameter 0 is dan wil je experimenteren met een zelf gedefinieerde doos.</p>
<p><b>PASCAL</b>  <code>function throwBall(holeIn, sideIn: integer; var holeOut, sideOut: integer): longint;</code></p> <p><b>C</b>  <code>int throwBall(int holeIn, int sideIn, int *holeOut, int *sideOut);</code></p> <p><b>C++</b>  <code>int throwBall(int holeIn, int sideIn, int &amp;holeOut, int &amp;sideOut);</code></p>	<p>Deze functie schiet een bal in het gat met nummer <code>holeIn</code> in de zijde <code>sideIn</code>. De zijdes zijn als volgt genummerd:</p> <ol style="list-style-type: none"> <li>1 – bovenzijde</li> <li>2 – rechterzijde</li> <li>3 – onderzijde</li> <li>4 – linkerzijde</li> </ol> <p>De gaten zijn oplopend genummerd van links naar rechts of van boven naar beneden, beginnend bij 1.</p> <p>In <code>holeOut</code> en <code>SideOut</code> krijg je te horen uit welk gat de bal weer tevoorschijn is gekomen.</p> <p>Het resultaat van de functie is een integer die het aantal piepjes aangeeft zodat je weet hoe vaak een kaatser door een bal is geraakt.</p>
<p><b>PASCAL</b>  <code>procedure ResetBox;</code></p> <p><b>C/C++</b>  <code>void ResetBox();</code></p>	<p>Deze functie zet alle kaatsers terug naar hun beginstand.</p>
<p><b>PASCAL</b>  <code>procedure ToggleDeflectors;</code></p> <p><b>C/C++</b>  <code>void ToggleDeflectors();</code></p>	<p>Deze functie verandert de toestand van alle kaatsers in de doos.</p>



<p>PASCAL procedure Finalize;</p> <p>C/C++ void Finalize();</p>	<p>Deze functie beëindigt je experiment met de doos. Je moet de functie aanroepen voordat je je programma beëindigt.</p>
---	--

Om de library te gebruiken moet je het volgende doen:

FreePascal: In de taak directory vind je de bestanden `pbbplib.o` en `pbbplib.ppu`. Deze bestanden kun je op de volgende wijze gebruiken:

```
uses pbbplib;
```

In het bestand `pblackbox.pas` vind je een voorbeeld hoe je de library kunt gebruiken.

C: In de taak directory vind je de bestanden `cbplib.o` en `cbplib.h`. Deze bestanden kun je gebruiken door het volgende statement aan je programma toe te voegen:

```
#include "cbplib.h"
```

In het bestand `cblackbox.c` vind je een voorbeeld van hoe je de library kunt gebruiken. Om je programma te compileren moet je het volgende commando geven:

```
gcc -o yourprogram cbplib.o yourprogram.c
```

C++: In de taak directory vind je de bestanden `cppbplib.o` en `cppbplib.h`. Deze bestanden kun je gebruiken door het volgende statement aan je programma toe te voegen:

```
#include "cbplib.h"
```

In het bestand `cblackbox.c` vind je een voorbeeld van hoe je de library kunt gebruiken. Om je programma te compileren moet je het volgende commando geven:

```
g++ -o yourprogram cppbplib.o yourprogram.cpp
```

**;;Let op dat er nooit meer dan één programma gelijktijdig de library mag gebruiken!!**

### VOORBEELD

Het voorbeeld dat hoort bij de doos in de eerste figuur is als volgt:

Functieaanroep	Beschrijving
<code>Initialize(0);</code>	Als je er vanuit gaat dat je programma werkt met de doos uit het voorbeeld, is het resultaat van deze aanroep 5 om aan te geven $n=5$ .
<p>PASCAL <code>throwBall(3,4,holeOut,sideOut);</code></p> <p>C <code>throwBall(3,4,&amp;holeOut,&amp;sideOut);</code></p> <p>C++ <code>throwBall(3,4,holeOut,sideOut);</code></p>	Een bal wordt aan de linkerkant ( $sideIn=4$ ) in het derde gat ( $holeIn=3$ ) van boven gegooid. Het resultaat van deze functie is 1 waarmee aangegeven wordt dat de bal één kaatser raakt. Als de functie aangeroepen is dan zal $sideOut=3$ en $holeOut=2$ zijn. Hiermee wordt aangegeven dat de bal aan de onderkant in het tweede gat van links de doos verlaten heeft.

### EXPERIMENTEREN

Je kunt zelf met de library experimenteren. Dat doe je door de `Initialize` met  $box = 0$  aan te roepen. De library leest dan de binnenkant van de zwarte doos uit het bestand `blackbox.in`. Het formaat van `blackbox.in` is als volgt:



<code>blackbox.in</code>	Beschrijving
5 3 2 3 \ 4 2 / 4 4 /	<p>Op de eerste regel staat een getal <math>n</math> dat aangeeft hoeveel gaten er in elke zijde zitten.</p> <p>Op de tweede regel staat een integer die aangeeft hoeveel kaatsers er in de doos zitten.</p> <p>Op de volgende regels staat telkens de positie en initiële stand van één kaatser beschreven door twee integers en een karakter, allen gescheiden door een spatie. Het eerste getal geeft aan in welke rij de kaatser staat. Het tweede getal geeft aan in welke kolom de kaatser staat. Het karakter is ' / ' of ' \ ' en geeft de beginstand van de kaatser aan.</p>

Dit voorbeeld beschrijft de figuur die bovenaan pagina 1 staat.

### ERROR MESSAGES

Als er iets misgaat in de library dan wordt er een foutboodschap afgedrukt naar standard error. De mogelijke foutboodschappen die je kunt krijgen, en de betekenis, staan in de volgende tabel:

Foutmelding	Betekenis
ERR 1 More than one app	Slechts één applicatie kan gelijktijdig met de black boxes experimenteren. Stop alle applicaties en start er maar één tegelijkertijd.
ERR 2 Invalid box	Het doosnummer dat je gebruikt ligt niet tussen 0 en 15.
ERR 3 Invalid deflector	Het bestand <code>blackbox.in</code> heeft een kaatser op een ongeldige positie.
ERR 4 Invalid symbol	Het bestand <code>blackbox.in</code> bevat een ongeldig symbool.
ERR 5 Invalid size	Het bestand <code>blackbox.in</code> heeft de verkeerde omvang.
ERR 6 Invalid input hole	Het nummer van de zijde of het invoergat is ongeldig.
ERR 7 ALARM	Roep onmiddellijk de technische staf!

### BEOORDELING

Voor elke doos die je krijgt moet je een tekstbestand inzenden dat zo goed mogelijk de binnenkant van de zwarte doos beschrijft. Per doos geldt het volgende:

- Als je programma een ' / ', ' \  
' of ' \  
' op de verkeerde plaats zet dan krijg je 0 punten voor die doos.
- Als  $B_m$  het maximale aantal posities is dat in een geldige inzending gevonden is, en jij hebt  $B_y$  correcte posities correct gevonden, dan is je procentuele score voor deze doos:

$$100 B_y / B_m$$

De officiële oplossing voor deze opdracht kan 100% van alle begintoestanden vinden binnen een tijd van 8 minuten per doos.