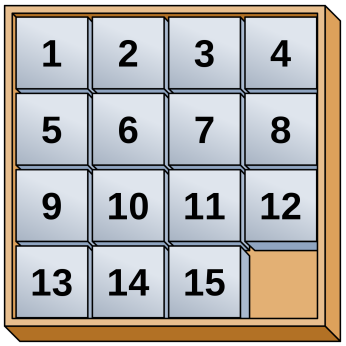
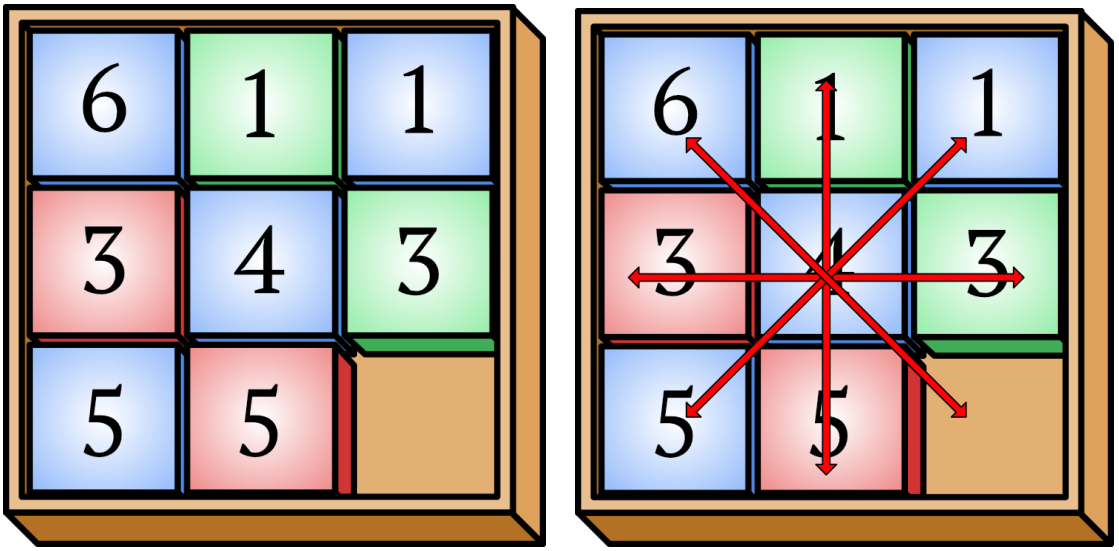
Un rompecabezas deslizante es un rompecabezas que reta al usuario a deslizar piezas (generalmente planas) a lo largo de rutas (normalmente sobre un tablero) para llegar a una configuración determinada final. Las piezas a mover pueden ser simples formas rectangulares o pueden tener colores, patrones, partes de una imagen mayor, números o letras.



Deseamos programar en Gobstones un puzzle de tales características, de tamaño fijo, 3x3, con una particularidad: las piezas poseen uno de tres colores (Azul, Rojo o Verde) y un número. El objetivo de nuestro puzzle es, contando desde el centro, la fila, la columna y ambas diagonales sumen el mismo número, y donde en la fila y en la columna no hay más de una pieza del misma color, pero en las diagonales todas las piezas son del mismo color.

A continuación un ejemplo de tal puzzle, en su estado de solución:



En el dibujo de la derecha se muestran la fila, la columna y las diagonales a considerar. Como se observa en la fila, la columna y las diagonales, los números suman 10. En la fila y la columna, no hay dos colores iguales, mientras que en las diagonales todos los elementos tienen el mismo color. Notar que se considera al espacio vacío que queda como el número cero y como cualquier color. Modelaremos este juego en un tablero de 3x3

Lo que se pide es que, considerando las siguientes funciones primitivas:

```
colorDeFichaAcá()
/*
    PROPÓSITO: Describe el color de la ficha en la celda actual.
         Sí en la celda no hubiera ficha describe Negro.
    TIPO: Color
    PRECONDICIONES: Ninguna.
*/
```

```
colorDeFichaAl_(dirección)
/*
    PROPÓSITO: Dada una dirección describe el color de la ficha en la
celda lindante en la dirección dada. Sí en la celda no hubiera ficha
describe Negro.
    PARÁMETROS: dirección - Dirección - La dirección de la celda lindante
        a mirar.
    TIPO: Color
    PRECONDICIONES: Hay al menos una celda en la dirección dada.
*/
```

```
colorDeFichaAl_YAl_(direcciónPrincipal, direcciónSecundaria)
/*
    PROPÓSITO: Dadas dos direcciones, describe el color de la ficha en la
celda en diagonal compuesta por las direcciones dadas. Sí en la celda no
hubiera ficha describe Negro.
    PARÁMETROS:
        * direcciónPrincipal - Dirección - La primera dirección de la
              diagonal de la celda a mirar.
        * direcciónSecundaria - Dirección - La segunda dirección de la
              diagonal de la celda a mirar.
    TIPO: Color
    PRECONDICIONES: Hay al menos una celda hacia cada una de las
        direcciones dadas. Las direcciones dadas no son opuestas ni
iguales.
*/
```

```
númeroDeFichaAcá()
/*
    PROPÓSITO: Describe el número de la ficha en la celda actual.
         Sí en la celda no hubiera ficha describe 0.
    TIPO: Número
    PRECONDICIONES: Ninguna.
*/
```

```
númeroDeFichaAl_(dirección)
/*
    PROPÓSITO: Dada una dirección describe el número de la ficha en la
celda lindante en la dirección dada. Sí en la celda no hubiera ficha
describe 0.
    PARÁMETROS: dirección - Dirección - La dirección de la celda lindante
        a mirar.
    TIPO: Número
    PRECONDICIONES: Hay al menos una celda en la dirección dada.
*/
```

```
númeroDeFichaAl_YAl_(direcciónPrincipal, direcciónSecundaria)
/*
    PROPÓSITO: Dadas dos direcciones, describe el número de la ficha en la
celda en diagonal compuesta por las direcciones dadas. Sí en la celda no
hubiera ficha describe 0.
    PARÁMETROS:
        * direcciónPrincipal - Dirección - La primera dirección de la
              diagonal de la celda a mirar.
        * direcciónSecundaria - Dirección - La segunda dirección de la
              diagonal de la celda a mirar.
    TIPO: Número
    PRECONDICIONES: Hay al menos una celda hacia cada una de las
        direcciones dadas. Las direcciones dadas no son opuestas ni
iguales.
*/
```

El objetivo consiste entonces en que escriba la función esConfiguraciónGanadora() que, suponiendo el cabezal en la celda central del tablero, y que en esa celda siempre hay una ficha para cualquier configuración ganadora, indica sí efectivamente la configuración dada es una configuración ganadora..

Como ayuda, se recomienda que piense subtareas desde aquellas cosas más grandes que tiene que hacer, hasta las más pequeñas. Recuerde que puede utilizar parámetros para generalizar su solución.