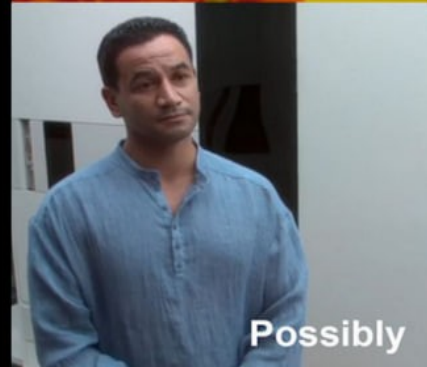


SPÉCIFICATION DU PROJET “Dames chinoises”  
*De Q1 à Q9 Édition*

## What's each language's stance on semicolons?



## Table des matières

1. Types	3
1.1 dimension	3
1.2 case	3
1.3 couleur	3
1.4 case_coloree	3
1.5 configuration	3
1.6 coup	4
1.7 vecteur	4
2. Fonctions de questions	5
2.1 indice_valide	5
2.2 est_case	5
2.3 est_dans_losange	6
2.4 est_dans_losange_2	7
2.5 est_dans_losange_3	8
2.6 est_dans_etoile	9
2.7 tourner_case	10
2.8 translate	11
2.9 diff_case	11
2.10 sont_cases_alignee	12
2.11 dist_coords	12
2.12 max_dist_cases	13
2.13 min_dist_cases	13
2.14 compte_cases	13
2.15 sont_cases_voisines	14
2.16 calcul_pivot	14
2.17 vec_et_dist	15
3. Fonctions de tests	16
3.1 randint	16
3.2 randtriplet	16
3.3 randcase	16
3.4 randvec	17
4. Fonctions d'affichage	18
4.1 string_of_couleur	18
4.2 transfo	18
4.3 associe (incomplet)	18
4.4 affiche_ligne	19
4.5 affiche	19

# 1. Types

## 1.1 dimension

SPÉCIFICATION – dimension	
PROFIL	$dimension \stackrel{\text{def}}{=} \mathbb{N}^*$
SÉMANTIQUE	$dimension$ est une dimension d'un plateau, noté $dim$ par la suite, est un paramètre qui encode la taille du plateau. Le plateau à $4dim+1$ lignes horizontales numérotées de bas en haut de $-2dim$ à $2dim$ et similairement pour les lignes obliques.

## 1.2 case

SPÉCIFICATION – case	
PROFIL	$case \stackrel{\text{def}}{=} \{(i, j, k) \in \mathbb{Z}^3 \text{ tel que } i+j+k=0\}$
SÉMANTIQUE	<p><math>case</math> est définie par trois coordonnées <math>(i, j, k)</math>, la case au centre du plateau de jeu a pour coordonnées <math>(0, 0, 0)</math>. Les coordonnées représentent:</p> <ul style="list-style-type: none"> <li>• <math>i</math> le numéro de la ligne horizontale ;</li> <li>• <math>j</math> le numéro de la ligne horizontale lorsqu'on a tourné le plateau d'un tiers de tour dans le sens anti-horaire ;</li> <li>• <math>k</math> le numéro de la ligne horizontale lorsqu'on a tourné le plateau d'un tiers de tour dans le sens horaire.</li> </ul>

## 1.3 couleur

SPÉCIFICATION – couleur	
PROFIL	$couleur \stackrel{\text{def}}{=} \{Vert, Jaune, Rouge, Noir, Bleu, Maron\} \cup$ $\{Libre\} \cup$ $\{Code(nom) \text{ tel que } nom \in string \text{ et } (String.length\ nom)=3\}$
SÉMANTIQUE	$couleur$ c'est les couleurs des joueurs. Le constructeur $Code$ permet d'entrer les noms de joueur restreint à trois caractères. La couleur $Libre$ est une couleur en plus pour coder l'absence de joueur (dans une case ou pour le gagnant d'une partie).

## 1.4 case\_coloree

SPÉCIFICATION – case colorée	
PROFIL	$case\_coloree \stackrel{\text{def}}{=} case \times couleur$
SÉMANTIQUE	$case\_coloree$ est un pion d'une couleur $col$ se situe sur une case $c$ est codé par un couple $(c, col)$ .

## 1.5 configuration

SPÉCIFICATION – configuration du jeu	
PROFIL	$configuration \stackrel{\text{def}}{=} case\_coloree\ list \times couleur\ list \times dimension$
SÉMANTIQUE	<p><math>configuration</math> du jeu est donnée par un triplet formé d'une liste de cases colorées, une liste de joueurs et une dimension. La liste de cases colorées donne l'emplacement des pions et leurs couleurs. On veillera à ce que pour chaque case <math>c</math> il y ait au plus un pion sur cette case, c'est-à-dire il y a au plus une couleur <math>col</math> tel que le couple <math>(c, col)</math> est dans la liste; l'absence de pion sur la case <math>c</math> sera codé par l'absence de couple <math>(c, col)</math> dans la liste et non pas avec <math>(c, Libre)</math>. La liste de joueur permet de savoir à qui est le tour (tête de liste) et quel sera le tour des suivants (en suivant l'ordre de la liste). Enfin même si elle ne change pas au cours de la partie la dimension <math>dim</math> est donnée dans la configuration car nous devons pouvoir accéder facilement à celle-ci et pouvoir en changer si nous</p>

souhaitons faire une partie sur un plateau de taille différente.

## 1.6 coup

### SPÉCIFICATION – coups unitaires et multiples

PROFIL  $\text{coup} \stackrel{\text{def}}{=} \{ Du(c1, c2) \text{ tel que } c1, c2 \in \text{case} \} \cup \{ Sm(cl) \text{ tel que } cl \in \text{case list} \}$

SÉMANTIQUE  $\text{coup}$  existe en de deux sortes:  
-  $Du$  pour les déplacements unitaires;  
-  $Sm$  pour les sauts multiples.

## 1.7 vecteur

### SPÉCIFICATION – vecteur

PROFIL  $\text{vecteur} \stackrel{\text{def}}{=} \text{case}$

SÉMANTIQUE  $\text{vecteur}$  est le synonyme de  $\text{case}$  comme un vecteur permettant des translation avec les même propriétés.

## 2. Fonctions de questions

### 2.1 indice\_valide

SPÉCIFICATION – indice_valide	
PROFIL	$\text{indice\_valide} : \mathbb{N} \longrightarrow \text{dimension} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{indice\_valide } x \text{ dim})$ vérifie si la coordonnée $x$ est valide dans la dimension $\text{dim}$ .
EX. ET PROP.	<p>1) <math>\text{indice\_valide}(x, \text{dim}) = \text{vrai}</math>;  <math>\forall x \in \mathbb{Z}, \forall \text{dim} \in \text{dimension}, -2 \text{dim} \leq x \leq 2 \text{dim}</math>.</p> <p>2) <math>\text{indice\_valide}(x, \text{dim}) = \text{faux}</math>;  <math>\forall x \in \mathbb{Z}, \forall \text{dim} \in \text{dimension}, -2 \text{dim} &lt; x \vee x &gt; 2 \text{dim}</math>.</p>

RÉALISATION – indice_valide	
ALGORITHME	par la composition booléenne suivante $-2 \text{dim} \leq x \leq 2 \text{dim}$ .
IMPLIMENT.	<pre>let indice_valide (x:int) (dim:dimension): bool =   -2 * dim &lt;= x &amp;&amp; x &lt;= 2 * dim ;;</pre>

### 2.2 est\_case

SPÉCIFICATION – est_case	
PROFIL	$\text{est\_case} : \text{case} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{est\_case } c)$ vérifie si $c$ est une case.
EX. ET PROP.	<p>1) <math>\text{est\_case}((i, j, k)) = \text{vrai}, \forall (i, j, k) \in \mathbb{Z}^3, i + j + k = 0</math>;</p> <p>2) <math>\text{est\_case}((i, j, k)) = \text{faux}, \forall (i, j, k) \in \mathbb{Z}^3, i + j + k \neq 0</math>.</p>

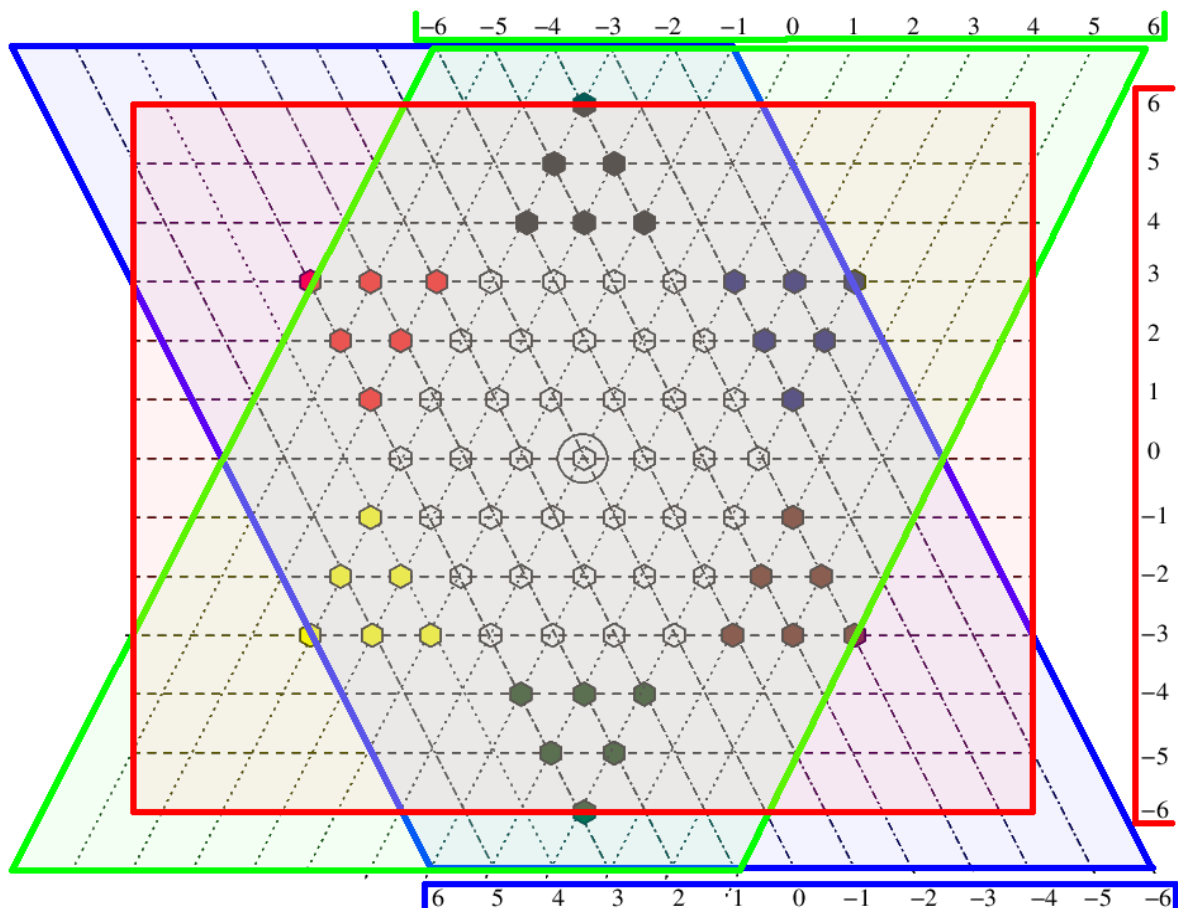


Figure 1. Représentation de la fonction *est\_case* par l'intersection de trois figure ou chacune c'est un des coordonnées

## RÉALISATION – est\_case

ALGORITHME composition booléenne si la somme des coordonnées  $i, j, k$  doit être égal à 0.

IMPLIMENT.

```
let est_case (c:case): bool =
  let i, j, k = c in i + j + k = 0
;;
```

## 2.3 est\_dans\_losange

## SPÉCIFICATION – est\_dans\_losange North-South

PROFIL  $\text{est\_dans\_losange} : \text{case} \longrightarrow \text{dimension} \longrightarrow \mathbb{B}$

SÉMANTIQUE ( $\text{est\_dans\_losange } c \text{ dim}$ ) vérifie si la case  $c$  est dans le losange North-South du plateau de dimension  $\text{dim}$ .

EX. ET PROP.

- 1)  $\text{est\_dans\_losange}((6, -3, -3), 3) = \text{vrai}$ ;
- 2)  $\text{est\_dans\_losange}((0, -3, 3), 3) = \text{vrai}$ ;
- 3)  $\text{est\_dans\_losange}((0, 3, -3), 3) = \text{vrai}$ ;
- 4)  $\text{est\_dans\_losange}((-6, 3, 3), 3) = \text{vrai}$ ;

$\text{est\_dans\_losange}((0,0,0), \text{dim}) = \text{vrai}, \forall \text{dim} \in \text{dimension}.$

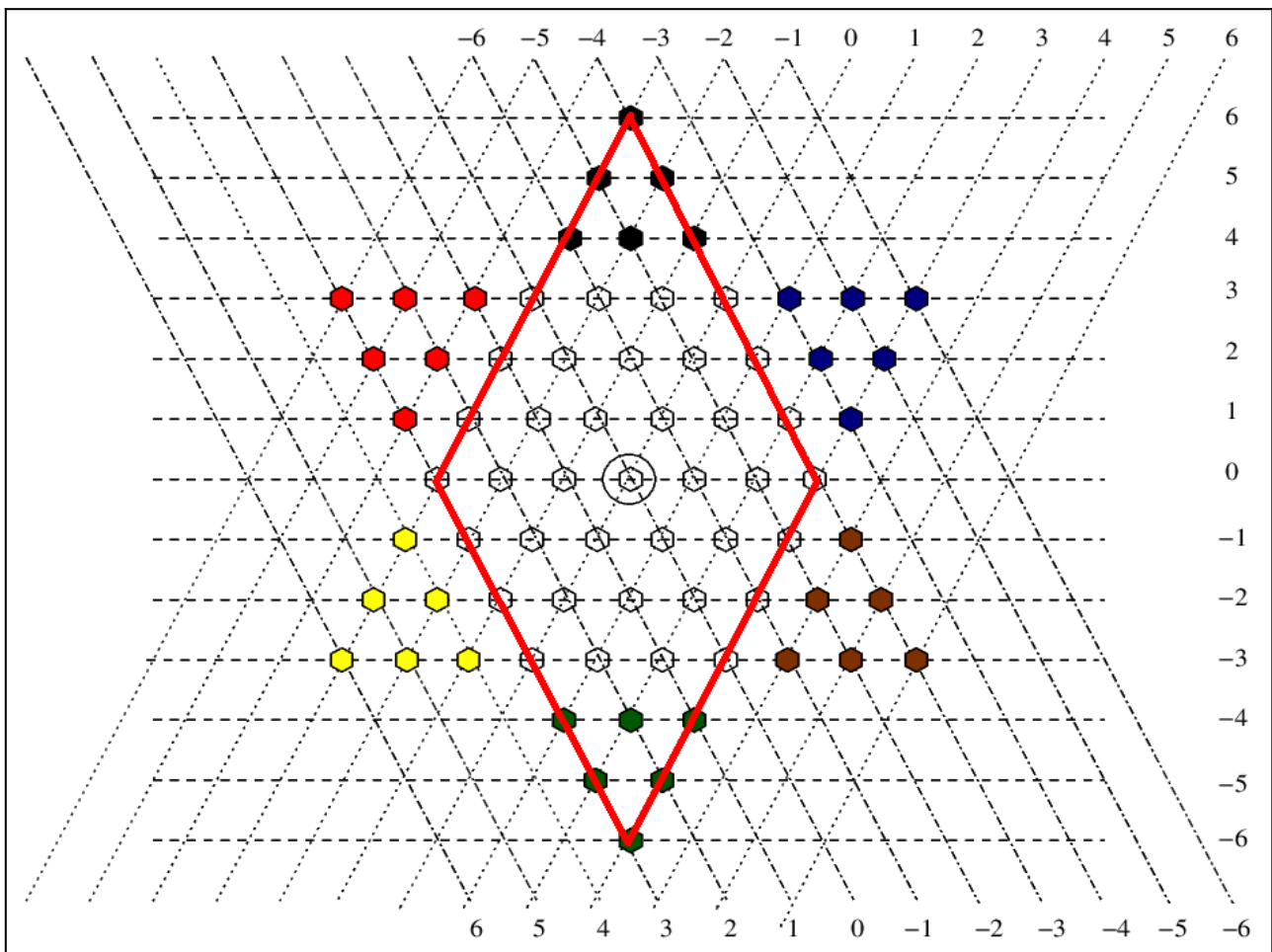


Figure 2. North-South losange

## RÉALISATION – est\_dans\_losange North-South

ALGORITHME composition booléenne si la case est dans le losange North-South de la figure 2.

IMPLIMENT.

```
let est_dans_losange (c:case) (dim:dimension): bool =
  let _, j, k = c in
    -dim <= j && j <= dim &&
    -dim <= k && k <= dim
;;
```

## 2.4 est\_dans\_losange\_2

SPÉCIFICATION – est_dans_losange_2 Northwest-Southeast	
PROFIL	$\text{est\_dans\_losange\_2} : \text{case} \longrightarrow \text{dimension} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{est\_dans\_losange\_2 } c \ dim)$ vérifie si la case $c$ est dans le losange Northwest-Southeast du plateau de dimension $dim$ .
EX. ET PROP.	<p>1) <math>\text{est\_dans\_losange\_2}((3, 3, -6), 3) = \text{vrai}</math>;            2) <math>\text{est\_dans\_losange\_2}((3, -3, 0), 3) = \text{vrai}</math>;            3) <math>\text{est\_dans\_losange\_2}((-3, -3, 6), 3) = \text{vrai}</math>;            4) <math>\text{est\_dans\_losange\_2}((-3, 3, 0), 3) = \text{vrai}</math>;</p> <p><math>(\text{est\_dans\_losange\_2 } (0,0,0), dim) = \text{vrai}, \forall dim \in \text{dimension}.</math></p>

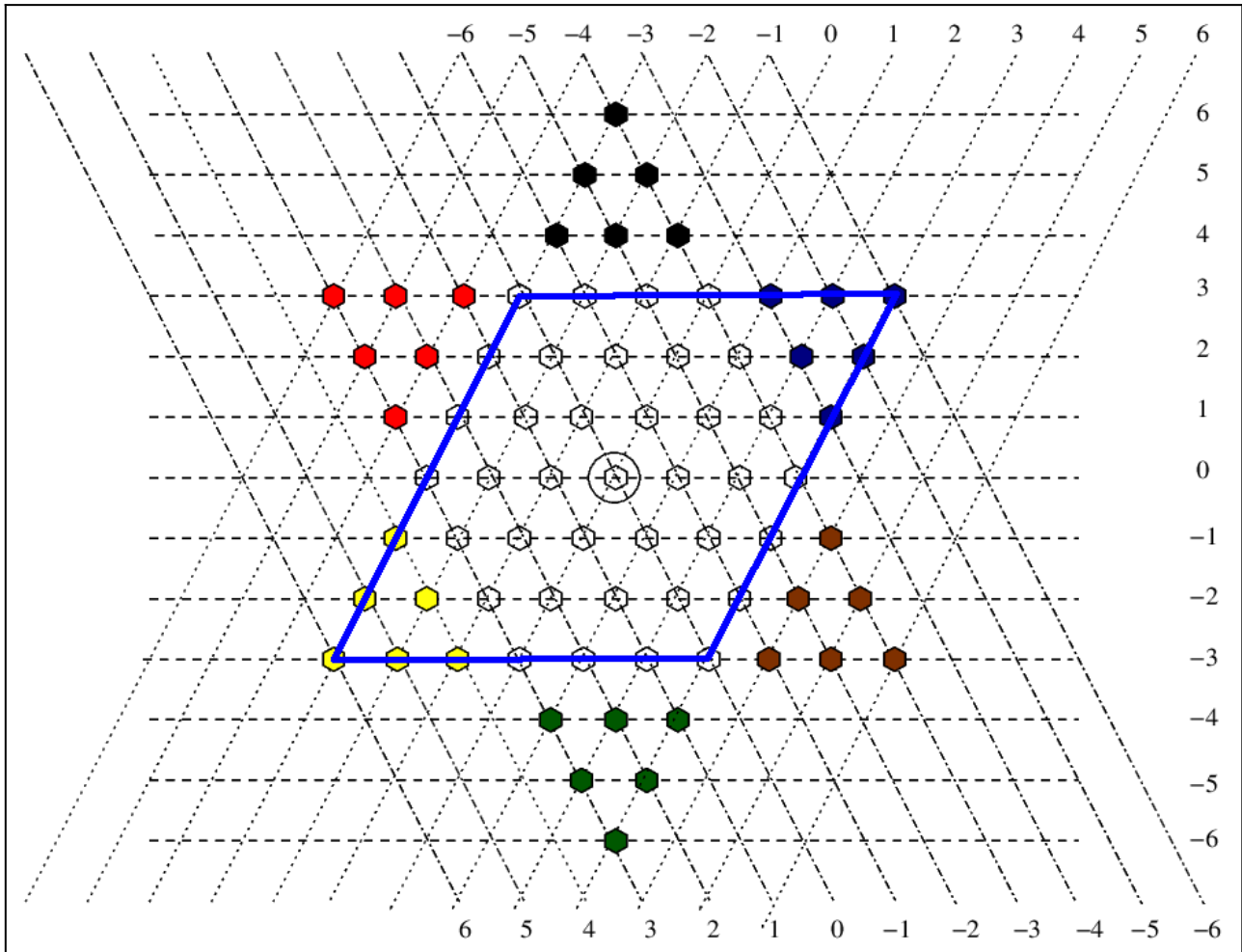


Figure 3. Northwest-Southeast losange

RÉALISATION – est_dans_losange_2 Northwest-Southeast	
ALGORITHME	composition booléenne si la case est dans le losange Northwest-Southeast de la figure 3.
IMPLIMENT.	<pre> let est_dans_losange_2 (c:case) (dim:dimension): bool =   let i, _, k = c in     -dim &lt;= i &amp;&amp; i &lt;= dim &amp;&amp;     -dim &lt;= k &amp;&amp; k &lt;= dim ;; </pre>



## 2.5 est\_dans\_losange\_3

SPÉCIFICATION – est_dans_losange_3 Northeast-Southwest	
PROFIL	$\text{est\_dans\_losange\_3} : \text{case} \longrightarrow \text{dimension} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{est\_dans\_losange\_3 } c \text{ } dim)$ vérifie si la case $c$ est dans le losange Northeast-Southwest du plateau de dimension $dim$ .
EX. ET PROP.	<p>1) <math>\text{est\_dans\_losange\_3}((3, -6, 3), 3) = \text{vrai}</math>;  2) <math>\text{est\_dans\_losange\_3}((3, 0, -3), 3) = \text{vrai}</math>;  3) <math>\text{est\_dans\_losange\_3}((-3, 6, -3), 3) = \text{vrai}</math>;  4) <math>\text{est\_dans\_losange\_3}((-3, 0, 3), 3) = \text{vrai}</math>;</p> <p><math>(\text{est\_dans\_losange\_3 } (0,0,0), dim) = \text{vrai}, \forall dim \in \text{dimension}.</math></p>

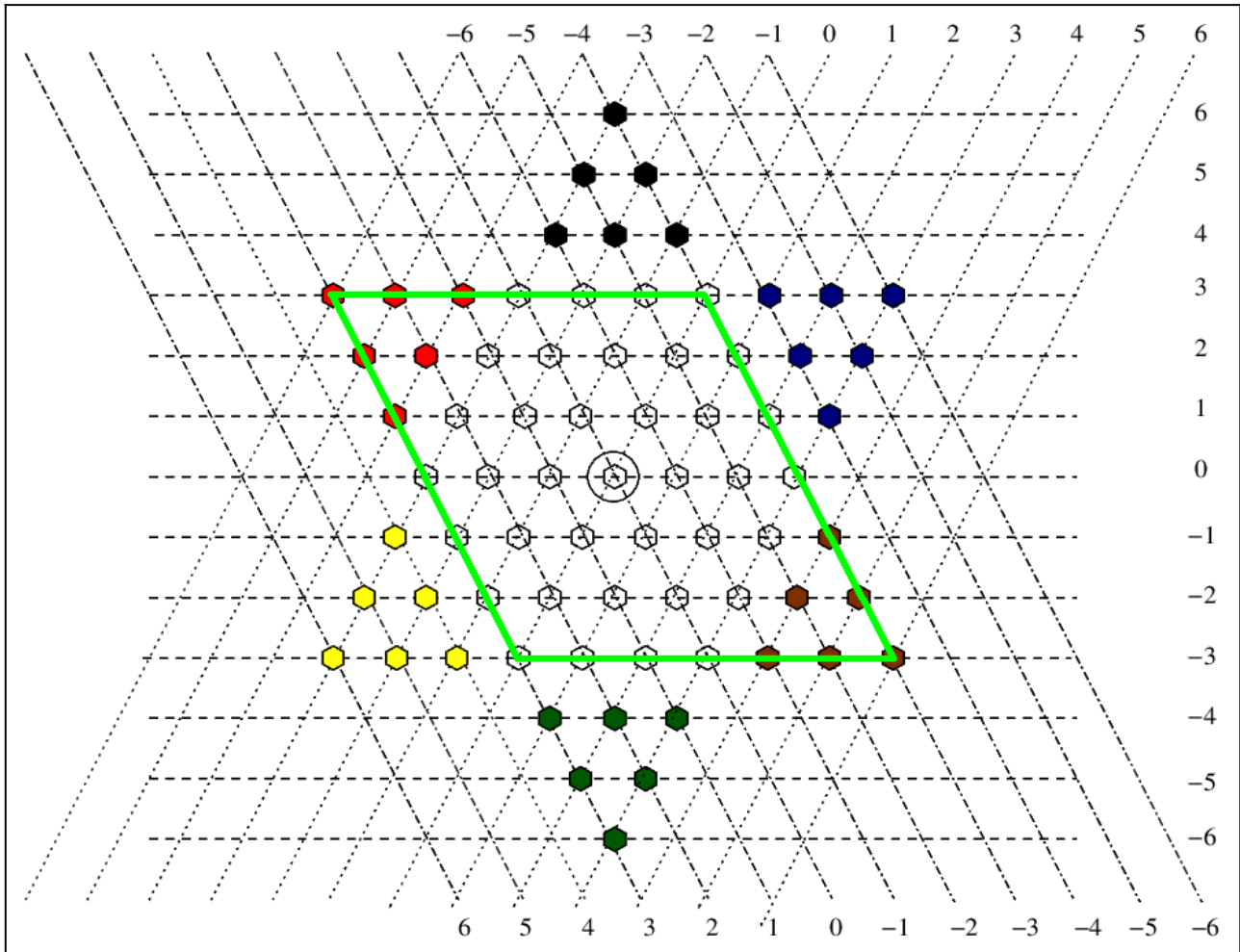


Figure 4. Northeast-Southwest losange

RÉALISATION – est_dans_losange_3 Northeast-Southwest	
ALGORITHME	composition booléenne si la case est dans le losange Northeast-Southwest de la figure 4.
IMPLIMENT.	<pre> let est_dans_losange_3 (c:case) (dim:dimension): bool =   let i, j, _ = c in     -dim &lt;= i &amp;&amp; i &lt;= dim &amp;&amp;     -dim &lt;= j &amp;&amp; j &lt;= dim ;; </pre>



## 2.6 est\_dans\_etoile

SPÉCIFICATION – est_dans_etoile	
PROFIL	$\text{est\_dans\_etoile} : \text{case} \longrightarrow \text{dimension} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{est\_dans\_etoile})$ vérifie si la case $c$ est dans l'étoile du plateau de dimension $\text{dim}$ .
EX. ET PROP.	<p>1) <math>\text{est\_dans\_etoile}((6, -3, -3), 3) = \text{vrai}</math>;  2) <math>\text{est\_dans\_etoile}((3, -6, 3), 3) = \text{vrai}</math>;  3) <math>\text{est\_dans\_etoile}((-3, -3, 6), 3) = \text{vrai}</math>;  4) <math>\text{est\_dans\_etoile}((-6, 3, 3), 3) = \text{vrai}</math>;  5) <math>\text{est\_dans\_etoile}((-3, 6, -3), 3) = \text{vrai}</math>;  6) <math>\text{est\_dans\_etoile}((3, 3, -6), 3) = \text{vrai}</math>;</p> <p><math>(\text{est\_dans\_etoile}(0,0,0), \text{dim}) = \text{vrai}, \forall \text{dim} \in \text{dimension}.</math></p>

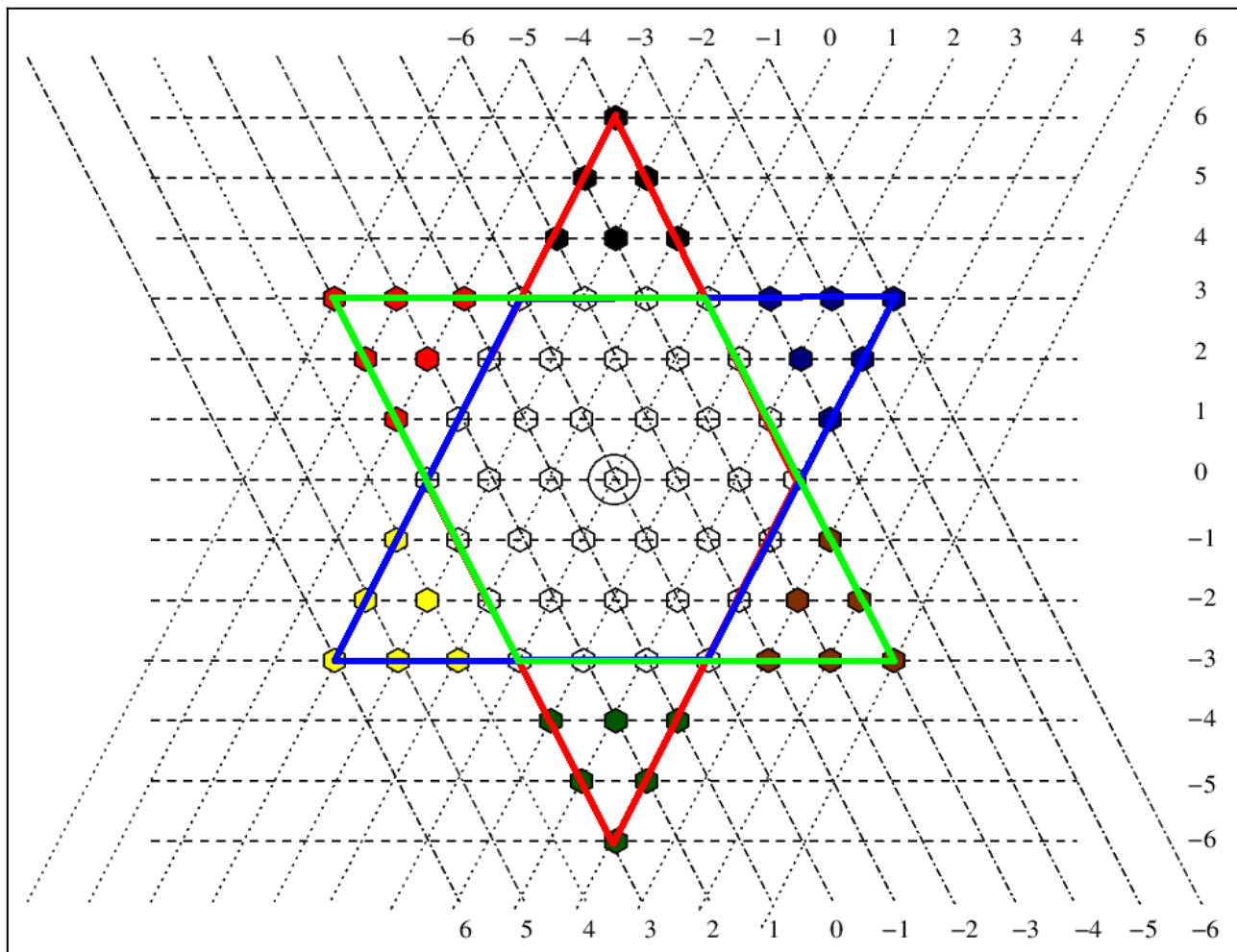


Figure 5. L'union de trois losanges forme une étoile

RÉALISATION – est_dans_etoile	
ALGORITHME	composition booléenne et fonctions si la case est dans l'union de trois losanges.
IMPLIMENT.	<pre> let est_dans_etoile (c:case) (dim:dimension): bool =   (* l'union de trois losange est un étoile *)   est_dans_losange c dim      est_dans_losange_2 c dim      est_dans_losange_3 c dim ;; </pre>

## 2.7 tourner\_case

SPÉCIFICATION – tourner_case	
PROFIL	$\text{tourner\_case} : \mathbb{N} \longrightarrow \text{case} \longrightarrow \text{case}$
SÉMANTIQUE	$(\text{tourner\_case } m \ c)$ c'est la case $c$ après avoir fait tourner le plateau de $m$ sixième de tour dans le sens anti-horaire.
EX. ET PROP.	<p>1) <math>\text{tourner\_case}(0, (4, -2, -2)) = (4, -2, -2)</math>;  2) <math>\text{tourner\_case}(1, (4, -2, -2)) = (2, -4, 2)</math>;  3) <math>\text{tourner\_case}(2, (4, -2, -2)) = (-2, -2, 4)</math>;  4) <math>\text{tourner\_case}(3, (4, -2, -2)) = (-4, 2, 2)</math>;  5) <math>\text{tourner\_case}(4, (4, -2, -2)) = (-2, 4, -2)</math>;  6) <math>\text{tourner\_case}(5, (4, -2, -2)) = (2, 2, -4)</math>;  7) <math>\text{tourner\_case}(6, (4, -2, -2)) = (4, -2, -2)</math>;</p> <p><math>\forall n \in \mathbb{N}, \forall (i, j, k) \in \text{case}, m = n \bmod 6,</math>  <math>\text{sim} = 0, \text{tourner\_case}(m, (i, j, k)) = (i, j, k)</math> .  <math>\text{sim} \neq 0, \text{tourner\_case}(m, (i, j, k)) = \text{tourner\_case}((m-1), (-k, -i, -j))</math></p>

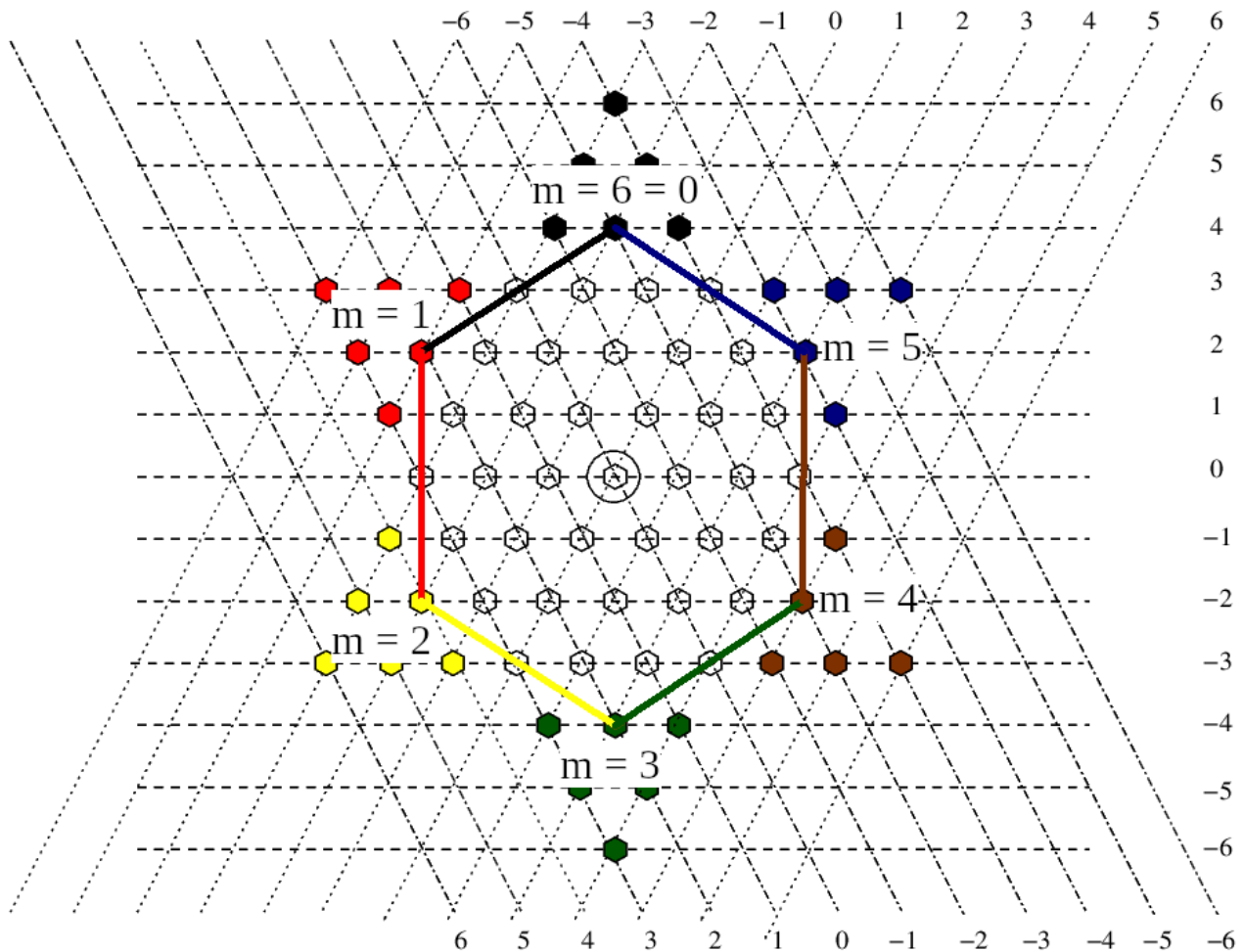


Figure 5. La case après avoir tourner 6 fois

RÉALISATION – tourner case	
On va utiliser une fonction locale <i>tourner</i> avec une équation de récurrence :	
(1) $\text{tourner}(0, (i, j, k)) = (i, j, k)$ (2) $\text{tourner}(m, (i, j, k)) = \text{tourner}((m-1), (-k, -i, -j))$	
ALGORITHME	composition conditionnelle, composition fonction et analyse par cas de filtrage.
IMPLIMENT.	<pre>let tourner_case (m:int) (c:case): case =   (* réduction de nombre de fait pour tourner *)   let m = m mod 6 in</pre>

```

(* équation réursive *)
let rec tourner_case_rec (m:int) (c:case): case =
  let i, j, k = c in
  match m with
  | 0 -> i, j, k
  | m -> tourner_case_rec (m - 1) (-k, -i, -j)
in tourner_case_rec m c
;;

```

## 2.8 translate

SPÉCIFICATION – translate	
PROFIL	$\text{translate} : \text{case} \longrightarrow \text{vecteur} \longrightarrow \text{case}$
SÉMANTIQUE	$(\text{translate } c \ v)$ calcule la case par le translation du vecteur $v$ à partir de $c$ .
EX. ET PROP.	<p>1) <math>\text{translate}((0, -2, 2), (0, 1, -1)) = (0, -1, 1);</math>            2) <math>\text{translate}((0, 0, 0), (0, -2, 2)) = (0, -2, 2);</math>            3) <math>\text{translate}((0, -2, 2), (0, 0, 0)) = (0, -2, 2);</math>            4) <math>\text{translate}((0, 0, 0), (0, 0, 0)) = (0, 0, 0);</math></p> <p><math>\forall (c_1, c_2, c_3), (v_1, v_2, v_3) \in \text{case}, (c_1 + v_1, c_2 + v_2, c_3 + v_3) \in \text{case}.</math></p>

RÉALISATION – translate	
ALGORITHME	addition de coordonnées de case $c$ et vecteur $v$ .
IMPLIMENT.	<pre> let translate (c:case) (v:vecteur): case =   (* les coordonnées de la case c *)   let c1, c2, c3 = c   (* les coordonnées du vecteur v *)   and v1, v2, v3 = v in   (* translation des coordonnées de v vers c *)   v1 + c1, v2 + c2, v3 + c3 ;; </pre>

## 2.9 diff\_case

SPÉCIFICATION – diff_case	
PROFIL	$\text{diff\_case} : \text{case} \longrightarrow \text{case} \longrightarrow \text{vecteur}$
SÉMANTIQUE	$(\text{diff\_case } c_1 \ c_2)$ c'est le vecteur de translation de $c_2$ vers $c_1$ , calculer par la différence entre les cases $c_1$ et $c_2$ .
EX. ET PROP.	<p>1) <math>\text{diff\_case}((0, -2, 2), (0, 1, -1)) = (0, -3, 3);</math>            2) <math>\text{diff\_case}((0, 0, 0), (0, -2, 2)) = (0, 2, -2);</math>            3) <math>\text{diff\_case}((0, -2, 2), (0, 0, 0)) = (0, -2, 2);</math>            4) <math>\text{diff\_case}((0, 0, 0), (0, 0, 0)) = (0, 0, 0);</math></p> <p><math>\forall (i_1, j_1, k_1), (i_2, j_2, k_2) \in \text{case}, (i_1 - i_2, j_1 - j_2, k_1 - k_2) \in \text{vecteur}.</math></p>

RÉALISATION – diff_case	
ALGORITHME	différence entre les coordonnées des cases $c_1$ et $c_2$ .
IMPLIMENT.	<pre> let diff_case (c1:case) (c2:case): vecteur =   let i1, j1, k1 = c1   and i2, j2, k2 = c2 in   (* la différence entre les coordonnées c1 et c2 *)   i1 - i2, j1 - j2, k1 - k2 ;; </pre>

## 2.10 sont\_cases\_alignee

SPÉCIFICATION – sont_cases_alignee	
PROFIL	$\text{sont\_cases\_alignee} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{sont\_cases\_alignee } c1 \ c2)$ vérifie si les cases $c1$ et $c2$ sont alignées.
EX. ET PROP.	<p>1) <math>\text{sont\_cases\_alignee}((0, -2, 2), (0, 2, -2)) = \text{vrai}</math>;  2) <math>\text{sont\_cases\_alignee}((-2, 0, 2), (2, 0, -2)) = \text{vrai}</math>;  3) <math>\text{sont\_cases\_alignee}((-2, 2, 0), (2, -2, 0)) = \text{vrai}</math>;</p> <p>4) <math>\text{sont\_cases\_alignee}((-2, 2, 0), (2, 0, -2)) = \text{faux}</math>;  pas de coordonnées qui est égal à l'autre.  <math>\forall c \in \text{case}, \text{sont\_cases\_alignee}(c, c) = \text{faux}</math>;  on effet c'est vrai, car chaque de coordonnées est égal, mais on idée les cases doivent être de différentes valeurs, donc c'est faux.</p>

RÉALISATION – sont_cases_alignee	
ALGORITHME	analyse par cas de filtrage
IMPLIMENT.	<pre> let sont_cases_alignee (c1:case) (c2:case): bool =   let i1, j1, k1 = c1   and i2, j2, k2 = c2 in   match () with (* comparasion entre chaque de coordonnées *)     (* les doivent être de différentes valeurs *)       _ when c1 = c2 -&gt; false     (* s'ils sont alignées sur i, donc vrai *)       _ when i1 = i2 -&gt; true     (* s'ils sont alignées sur j, donc vrai *)       _ when j1 = j2 -&gt; true     (* s'ils sont alignées sur k, donc vrai *)       _ when k1 = k2 -&gt; true       _ -&gt; false (* si les cases ne sont pas alignées *) ;; </pre>

## 2.11 dist\_coords

SPÉCIFICATION – dist_coords	
PROFIL	$\text{dist\_coords} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{N}^3$
SÉMANTIQUE	$(\text{dist\_coords } c1 \ c2)$ est un triplet de distances entre les coordonnées des cases $c1$ et $c2$ .
EX. ET PROP.,	<p>1) <math>\text{dist\_coords}((0, 0, 0), (0, -2, 2)) = (0, 2, 2)</math>;  2) <math>\text{dist\_coords}((0, 2, -2), (0, 0, 0)) = (0, 2, 2)</math>;  3) <math>\text{dist\_coords}((0, 2, -2), (0, -2, 2)) = (0, 4, 4)</math>;</p> <p><math>\forall c \in \text{case}, \text{dist\_coords}(c, c) = (0, 0, 0)</math>;</p>

RÉALISATION – dist_coords	
ALGORITHME	calcul de chaque coordonnées utilisent la fonction <i>abs</i> .
IMPLIMENT.	<pre> let dist_coords (c1:case) (c2:case): int * int * int =   let i1, j1, k1 = c1   and i2, j2, k2 = c2 in   let di = abs (i1 - i2) (* distance entre les coordonnées i *)   and dj = abs (j1 - j2) (* distance entre les coordonnées j *)   and dk = abs (k1 - k2) (* distance entre les coordonnées k *)   in di, dj, dk (* triplet des distances entres i, j et k *) ;; </pre>

## 2.12 max\_dist\_cases

SPÉCIFICATION – max_dist_cases	
PROFIL	$\text{max\_dist\_cases} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{N}$
SÉMANTIQUE	$(\text{max\_dist\_cases } c1 \ c2)$ est la distance maximale entre les coordonnées des cases $c1$ et $c2$ .
EX. ET PROP.	<p>1) <math>\text{max\_dist\_cases}((0, 0, 0), (0, -2, 2)) = 2;</math>  2) <math>\text{max\_dist\_cases}((0, 2, -2), (0, 0, 0)) = 2;</math>  3) <math>\text{max\_dist\_cases}((0, 2, -2), (0, -2, 2)) = 4;</math></p> <p><math>\forall c \in \text{case}, \text{dist\_coords}(c, c) = 0.</math></p>

RÉALISATION – max_dist_cases	
ALGORITHME	utilisation des fonctions <code>dist_coords</code> et <code>max</code> .
IMPLIMENT.	<pre>let max_dist_cases (c1:case) (c2:case): int =   let di, dj, dk = dist_coords c1 c2 in   max di (max dj dk) ;;</pre>

## 2.13 min\_dist\_cases

SPÉCIFICATION – min_dist_cases	
PROFIL	$\text{min\_dist\_cases} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{N}$
SÉMANTIQUE	$(\text{min\_dist\_cases } c1 \ c2)$ est la distance minimale entre les coordonnées des cases $c1$ et $c2$ .
EX. ET PROP.	<p>1) <math>\text{min\_dist\_cases}((0, 0, 0), (0, -2, 2)) = 0;</math>  2) <math>\text{min\_dist\_cases}((-2, 3, -1), (0, 0, 0)) = 1;</math>  3) <math>\text{min\_dist\_cases}((0, 3, -3), (-2, -2, 4)) = 2;</math>  3) <math>\text{min\_dist\_cases}((0, 3, -3), (-3, -3, 6)) = 3;</math></p> <p><math>\forall c \in \text{case}, \text{dist\_coords}(c, c) = 0.</math></p>

RÉALISATION – min_dist_cases	
ALGORITHME	utilisation des fonctions <code>dist_coords</code> et <code>min</code> .
IMPLIMENT.	<pre>let min_dist_cases (c1:case) (c2:case): int =   let di, dj, dk = dist_coords c1 c2 in   min di (min dj dk) ;;</pre>

## 2.14 compte\_cases

SPÉCIFICATION – compte_cases	
PROFIL	$\text{compte\_cases} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{N}$
SÉMANTIQUE	$(\text{compte\_cases } c1 \ c2)$ est le nombre de cases entres les cases $c1$ et $c2$ . Pour déterminer ce nombre on prend la distance maximale si ils sont alignées, sinon la distance minimale.
EX. ET PROP.	<p>1) <math>\text{compte\_cases}((0, -1, 1), (0, 1, -1)) = 1;</math>  2) <math>\text{compte\_cases}((2, -3, 1), (2, 0, -2)) = 2;</math>  3) <math>\text{compte\_cases}((0, -2, 2), (0, 2, -2)) = 3;</math>  4) <math>\text{compte\_cases}((3, -3, 0), (-2, 2, 0)) = 4;</math>  5) <math>\text{compte\_cases}((0, -3, 3), (0, 3, -3)) = 5;</math></p> <p><math>\forall c \in \text{case}, \text{compte\_cases}(c, c) = 0.</math></p>

RÉALISATION – compte_cases	
ALGORITHME	analyse par cas de filtrage et utilisation des fonctions <i>sont_cases_alignee</i> , <i>max_dist_cases</i> et <i>min_dist_cases</i> .
IMPLIMENT.	<pre> let compte_cases (c1:case) (c2:case): int =   match () with   (* si les cases sont égal *)     _ when c1 = c2 -&gt; 0   (* si les cases sont alignées *)     _ when sont_cases_alignee c1 c2 -&gt; max_dist_cases c1 c2 - 1   (* sinon ... *)     _ -&gt; min_dist_cases c1 c2 - 1 ;; </pre>

## 2.15 sont\_cases\_voisines

SPÉCIFICATION – sont_cases_voisines	
PROFIL	$\text{sont\_cases\_voisines} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{sont\_cases\_voisines } c1 \ c2)$ vérifie si les cases $c1$ et $c2$ sont voisines.
EX. ET PROP.	<p>1) <math>\text{sont\_cases\_voisines}((0, 0, 0), (0, -1, 1)) = \text{vrai}</math>;  2) <math>\text{sont\_cases\_voisines}((0, 0, 0), (0, 1, -1)) = \text{vrai}</math>;  3) <math>\text{sont\_cases\_voisines}((0, 0, 0), (-1, 0, 1)) = \text{vrai}</math>;</p> <p><math>\forall c \in \text{case}, \text{sont\_cases\_voisines}(c, c) = \text{faux}</math>.</p>

RÉALISATION – sont_cases_voisines	
ALGORITHME	composition booléenne et utilisation des fonctions <i>sont_cases_alignee</i> et <i>max_dist_cases</i> .
IMPLIMENT.	<pre> let sont_cases_voisines (c1:case) (c2:case): bool =   (* si les cases sont alignées et la distances entre eux est 1 *)   sont_cases_alignee c1 c2 &amp;&amp; max_dist_cases c1 c2 = 1 ;; </pre>

## 2.16 calcul\_pivot

SPÉCIFICATION – calcul_pivot	
PROFIL	$\text{calcul\_pivot} : \text{case} \longrightarrow \text{case} \longrightarrow \text{case option}$
SÉMANTIQUE	$(\text{calcul\_pivot } c1 \ c2)$ calcul le pivot entre les cases $c1$ et $c2$ si ils sont alignées et le nombre de cases entre les deux est impair, sinon <i>None</i> .
EX. ET PROP.	<p>1) <math>\text{calcul\_pivot}((0, -1, 1), (0, 1, -1)) = \text{Some } (0, 0, 0)</math>;  2) <math>\text{calcul\_pivot}((1, 0, -1), (-1, 0, 1)) = \text{Some } (0, 0, 0)</math>;  3) <math>\text{calcul\_pivot}((-1, 1, 0), (1, -1, 0)) = \text{Some } (0, 0, 0)</math>;  4) <math>\text{calcul\_pivot } (0, -2, 2) \ (2, 0, -2) = \text{None}</math>;  5) <math>\text{calcul\_pivot } (2, 0, -2) \ (-2, 2, 0) = \text{None}</math>;  6) <math>\text{calcul\_pivot } (-2, 2, 0) \ (0, -2, 2) = \text{None}</math>;</p> <p><math>\forall c \in \text{case}, \text{calcul\_pivot}(c, c) = \text{None}</math>.</p>

RÉALISATION – calcul_pivot	
ALGORITHME	composition conditionnelle et utilisation de fonctions <i>mod</i> , <i>compte_cases</i> , <i>translate</i> et <i>sont_cases_alignee</i> .
IMPLIMENT.	<pre> let calcul_pivot (c1:case) (c2:case): case option =   (* si le nombre de cases entre c1 et c2 est impair *)   let est_impair = (compte_cases c1 c2) mod 2 = 1   (* les coordonnées du vecteur de translation de c2 vers c1 *)   and i, j, k = diff_case c1 c2 in   (* le vecteur de translation de c2 vers le mi-chemin de c1 *) </pre>

```

let v = i/2, j/2, k/2 in
(* les coordonnées de pivot *)
let p = translate c2 v in
if est_impair && sont_cases_alignee c1 c2
then Some(p) (* si impair et alignées, pivot existe *)
else None    (* sinon, pivot n'existe pas *)
;;

```

## 2.17 vec\_et\_dist

SPÉCIFICATION – vec_et_dist	
PROFIL	$\text{vec\_et\_dist} : \text{case} \longrightarrow \text{case} \longrightarrow \text{vecteur} \times \mathbb{N}$
SÉMANTIQUE	$(\text{vec\_et\_dist } c1 \ c2)$ est le couple $(v, d)$ avec $v$ le vecteur de translation d'un déplacement unitaire des cases alignées $c1$ vers $c2$ et avec $d$ la distance entre $c1$ et $c2$ . Si le vecteur unitaire n'existe pas, alors on renvoie $((0, 0, 0), 0)$ .
EX. ET PROP.	1) $\text{vec\_et\_dist}((0, -2, 2), (0, 0, 0)) = ((0, 1, -1), 2)$ ; 2) $\text{vec\_et\_dist}((0, 0, 0), (0, -2, 2)) = ((0, -1, 1), 2)$ ; 3) $\text{vec\_et\_dist}((0, -2, 2), (0, -2, 2)) = (0, 0, 0, 0)$ ; 4) $\text{vec\_et\_dist}((0, -2, 2), (-2, 2, 0)) = (0, 0, 0, 0)$ ;  $\text{vec\_et\_dist}(c, c) = ((0, 0, 0), 0), \forall c \in \text{case}$ .

RÉALISATION – vec_et_dist	
ALGORITHME	composition conditionnelle et utilisation des fonctions <i>sont_cases_alignee</i> , <i>max_dist_cases</i> et <i>diff_case</i> .
IMPLIMENT.	<pre> let vec_et_dist (c1:case) (c2:case): vecteur * int = (* si c1 = c2 ou non alignées renvoie nuls *) if c1 = c2    not (sont_cases_alignee c1 c2) then (0, 0, 0), 0 else (* sinon ... *) (* la distance entre les cases *) let d = max_dist_cases c1 c2 (* les coordonnées du vecteur de translation de c2 vers c1 *) and i, j, k = diff_case c1 c2 in (* les coordonnées du vecteur de translation unitaire de *) (* c2 vers c1 *) let i, j, k = i/d, j/d, k/d in (* le vecteur de translation unitaire de c1 vers c2 *) let v = i * (-1), j * (-1), k * (-1) in if est_case v then v, d (* si c'est un vecteur *) else (0, 0, 0), 0 (* sinon *) ;; </pre>



## 3. Fonctions de tests

### 3.1 randint

SPÉCIFICATION – randint	
PROFIL	$\text{randint} : \mathbb{N} \longrightarrow \mathbb{N} \longrightarrow \mathbb{N}$
SÉMANTIQUE	$(\text{randint } a \ b)$ renvoie un nombre entier aléatoire dans l'intervalle $[a, b]$ .
EX. ET PROP.	$\forall n, a, b \in \mathbb{N}, n - a \leq \text{randint}(n - a, n + b) \leq n + b$ $\forall n \in \mathbb{N}, \text{randint}(n, n) = n$

RÉALISATION – randint	
ALGORITHME	utilisation de module <i>Random</i> avec la fonction <i>int</i> .
IMPLIMENT.	<pre>let randint (a:int) (b:int): int =   Random.int (b - a + 1) + a ;;</pre>

### 3.2 randtriplet

SPÉCIFICATION – randtriplet	
PROFIL	$\text{randtriplet} : \mathbb{N} \longrightarrow \mathbb{N} \longrightarrow \mathbb{N}^3$
SÉMANTIQUE	$(\text{randtriplet } a \ b)$ renvoie un triplet entier aléatoire dans l'intervalle $[a, b] \times [a, b] \times [a, b]$ .
EX. ET PROP.	$\forall n, a, b \in \mathbb{N}, (n - a, n - a, n - a) \leq \text{randtriplet}(a, b) \leq (n + b, n + b, n + b)$ $\forall n \in \mathbb{N}, \text{randtriplet}(n, n) = (n, n, n)$

RÉALISATION – randtriplet	
ALGORITHME	utilisation de fonction <i>randint</i> .
IMPLIMENT.	<pre>let randtriplet (a:int) (b:int): int * int * int =   let x = randint a b   and y = randint a b   and z = randint a b in x, y, z ;;</pre>

### 3.3 randcase

SPÉCIFICATION – randcase	
PROFIL	$\text{randcase} : \text{dimension} \longrightarrow \text{case}$
SÉMANTIQUE	$(\text{randcase } \text{dim})$ renvoie une case aléatoire non nuls dans l'étoile de dimension <i>dim</i> .
EX. ET PROP.	$\forall \text{dim} \in \text{dimension}, c = \text{randcase}(\text{dim}),$ $c \neq (0, 0, 0) \wedge \text{est\_case}(c) \wedge \text{est\_dans\_etoile}(c)$

RÉALISATION – randcase	
Équation de récurrence :	
$c = \text{randtriplet}(-2 \text{ dim}, 2 \text{ dim})$	
(1) $\text{randcase}(\text{dim}) = c$ , <span style="margin-left: 2em;"><math>\text{si } c \neq (0, 0, 0) \wedge \text{est\_case}(c) \wedge \text{est\_dans\_etoile}(c, \text{dim})</math></span> (2) $\text{randcase}(\text{dim}) = \text{randcase}(\text{dim})$ , sinon	
ALGORITHME	composition conditionnelle et utilisation des fonctions <i>randtriplet</i> et <i>est_case</i> , <i>est_dans_etoile</i> .
IMPLIMENT.	<pre>let rec randcase (dim:dimension): case =   let c = randtriplet (-dim * 2) (dim * 2) in   if c &lt;&gt; (0,0,0) &amp;&amp; est_case c &amp;&amp; est_dans_etoile c dim then c   else randcase dim ;;</pre>

### 3.4 randvec

SPÉCIFICATION – randvec	
PROFIL	randcase : <i>vecteur</i>
SÉMANTIQUE	( <i>randvec</i> ) renvoie un vecteur aléatoire non nuls de la taille 1.
EX. ET PROP.	$v = \text{randvec}(), v \neq (0, 0, 0) \wedge (-1, -1, -1) \leq v \leq (1, 1, 1)$
RÉALISATION – randvec	
ALGORITHME	utilisation de fonction <i>randcase</i> .
IMPLIMENT.	<pre>let randvec (): vecteur = randcase 1 ;;</pre>

## 4. Fonctions d'affichage

### 4.1 string\_of\_couleur

SPÉCIFICATION – string_of_couleur	
PROFIL	$\text{string\_of\_couleur} : \text{couleur} \longrightarrow \text{string}$
SÉMANTIQUE	$(\text{string\_of\_couleur } col)$ est une chaîne de caractères qui représente la couleur $col$ .
EX. ET PROP.	<p>1) <math>(\text{string\_of\_couleur}(\text{Libre}) = ".");</math>            2) <math>(\text{string\_of\_couleur}(\text{Vert}) = "V");</math>            3) <math>(\text{string\_of\_couleur}(\text{Jaune}) = "J");</math>            4) <math>(\text{string\_of\_couleur}(\text{Rouge}) = "R");</math>            5) <math>(\text{string\_of\_couleur}(\text{Noir}) = "N");</math>            6) <math>(\text{string\_of\_couleur}(\text{Bleu}) = "B");</math>            7) <math>(\text{string\_of\_couleur}(\text{Marron}) = "M").</math></p> <p><math>\forall \text{Code}(s) \in \text{couleur}, \text{string\_of\_couleur}(\text{Code}(s)) = s</math></p>

RÉALISATION – string_of_couleur	
ALGORITHME	analyse par cas de filtrage.
IMPLIMENT.	<pre>let string_of_couleur (col:couleur):string =   match col with     Libre   -&gt; " ."     Vert    -&gt; " V "     Jaune   -&gt; " J "     Rouge   -&gt; " R "     Noir    -&gt; " N "     Bleu    -&gt; " B "     Marron  -&gt; " M "     Code s  -&gt; s ;;</pre>

### 4.2 transfo

SPÉCIFICATION – transfo	
PROFIL	$\text{transfo} : \mathbb{N} \longrightarrow \mathbb{N} \longrightarrow \text{case}$
SÉMANTIQUE	$(\text{transfo } x \ y)$ transforme les coordonnées cartésiennes $(x, y)$ en coordonnées de triplet $(u, v, w)$ .
EX. ET PROP.	

RÉALISATION – transfo	
ALGORITHME	utilisation de opération mathématique
IMPLIMENT.	<pre>let transfo (x:int) (y:int): int * int * int =   let u = y (* transformation en u *)   and v = (x - y) / 2 (* transformation en v *)   and w = (-x - y) / 2 in (* transformation en w *)   u, v, w (* transformation de (x,y) en (u, v, w) *) ;;</pre>

### 4.3 associe (incomplet)

SPÉCIFICATION – associe (incomplet)	
PROFIL	$\text{associe} : \text{case} \longrightarrow \text{case\_coloree list} \longrightarrow \text{couleur} \longrightarrow \text{couleur}$
SÉMANTIQUE	$(\text{associe } c \ \text{lcc } \text{defaut})$ vaut $col$ si $(c, col)$ est dans la liste et $\text{defaut}$ sinon. Si plusieurs couple avec $c$ existe le premier est utilisé.
EX. ET PROP.	<i>aucun, car la fonction est incomplet</i>

RÉALISATION – associe (incomplet)	
ALGORITHME	<i>aucun, car la fonction est incomplet</i>
IMPLIMENT.	<pre> let associe c lcc default =   default ;; </pre>

#### 4.4 affiche\_ligne

SPÉCIFICATION – affiche_ligne	
PROFIL	$\text{affiche\_ligne} : \mathbb{N} \longrightarrow \mathbb{N} \longrightarrow \text{configuration} \longrightarrow \text{string}$
SÉMANTIQUE	$(\text{affiche\_ligne } n \ m \ \text{config})$ est une ligne de plateau du jeu de la coordonnée $i$ .
EX. ET PROP.	<i>compliquer a faire</i>

RÉALISATION – affiche_ligne	
<p>Équation de récurrence de la fonction <i>affiche_ligne_aux</i> :</p> <p>(1) <math>\text{affiche\_ligne\_aux}(n, 4 \cdot \text{dim} + 1) = "\square"</math></p> <p>(2) si <math>\neg((n+m) \bmod 2 = 0)</math> ou <math>\neg(\text{est\_dans\_etoile } c \ \text{dim})</math>,  <math>\text{affiche\_ligne\_aux}(n, m) = "\square" \wedge \text{affiche\_ligne\_aux}(n, (m+1))</math>,  sinon,  <math>\text{affiche\_ligne\_aux}(n, m) = (\text{string\_of\_couleur}(\text{associe } c \ \text{lcc Libre})) \wedge \text{affiche\_ligne\_aux}(n, (m+1))</math></p>	
ALGORITHME	composition par plusieurs expressions conditionnelle et utilisations de fonctions <i>est_dans_etoile</i> , <i>string_of_couleur</i> et <i>associe</i> .
IMPLIMENT.	<pre> let affiche_ligne n m config =   (* lcc est un case_coloree et dim est un dimension *)   let (lcc, _, dim) = config in   let rec affiche_ligne_aux (n:int) (m:int): string =     if m = 4 * dim + 1 then " " (* fin de ligne *)     else       (* transformation de coordonnees cartisien au case *)       let c = transfo m n in       (* ceci est une inter-case (case inutile d'un damier)       ou hors de l'etoile *)       if not ((n + m) mod 2 = 0)            not (est_dans_etoile c dim) then         (* une place vide *)         let nul = " "         (* concatenation d'une place vide avec les autres         chaines *)         and s = affiche_ligne_aux n (m + 1) in nul ^ s       else (* ceci est une case ou bien en dehors du plateau *)         (* col c'est la couleur obtenue par l'associe *)         let col = (string_of_couleur (associe c lcc Libre))         (* concatenation d'une couleur avec les autres         chaines *)         and s = affiche_ligne_aux n (m + 1) in col ^ s     in affiche_ligne_aux n m   ;; </pre>

#### 4.5 affiche (incomplet)

SPÉCIFICATION –	
PROFIL	$" \rightarrow "$
SÉMANTIQUE	
EX. ET PROP.	

RÉALISATION –
---------------

Équation de récurrence :

ALGORITHME

IMPLIMENT.