

Projet 1

1. Sujet 1 : Life & Death

 Énoncé

On considère une grille - théoriquement infinie - dont les cases appelées *cellules* peuvent prendre deux états distincts : «vivante» ou «morte».

Une cellule possède huit voisins : les cellules adjacentes horizontalement, verticalement et diagonalement.

À chaque génération, les cellules peuvent changer d'état ou conserver leur état selon les règles suivantes:

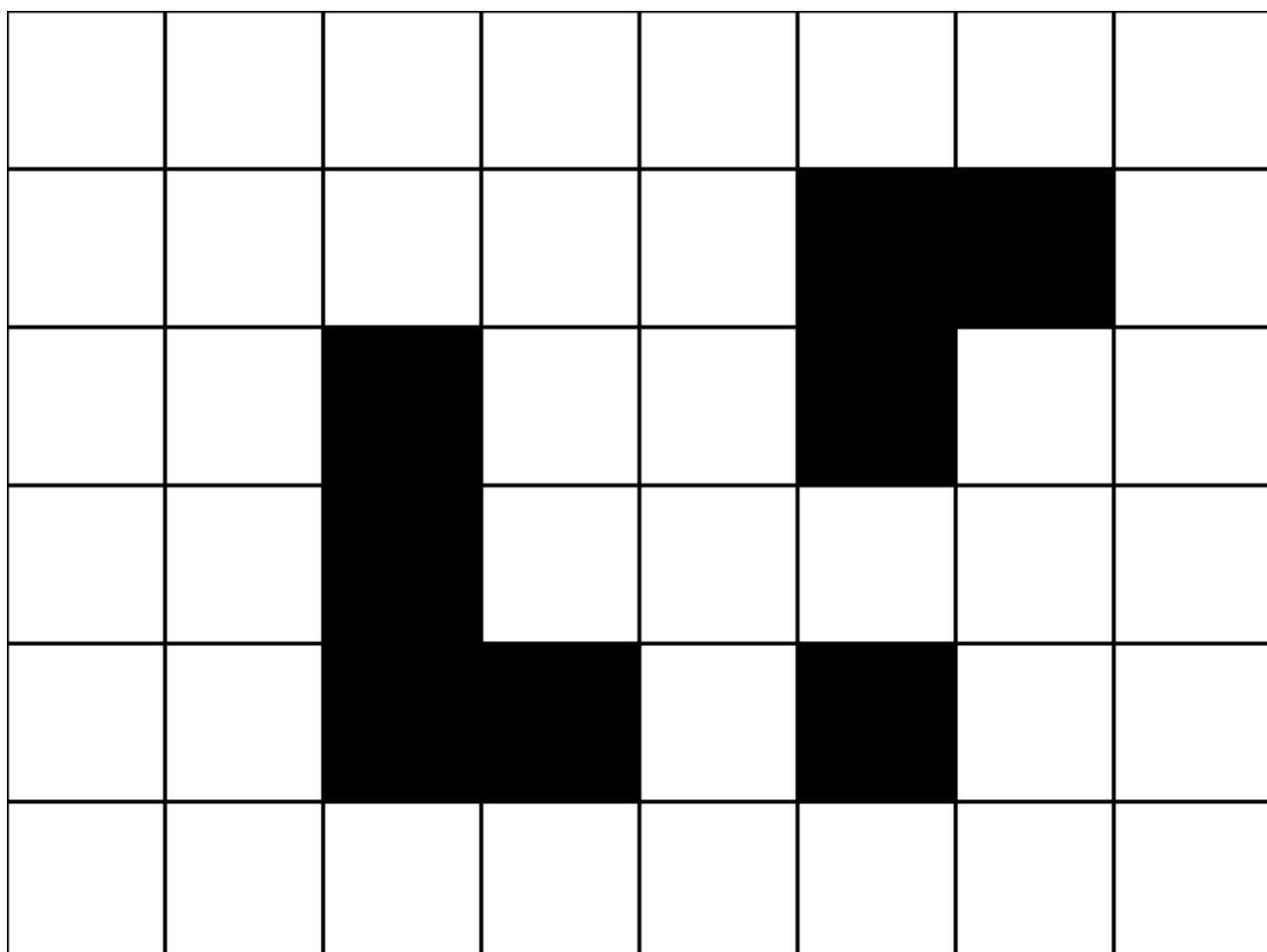
- si une cellule **morte** possède *exactement* 3 cellules voisines vivantes, alors elle devient **vivante** (elle naît);
- si une cellule **vivante** possède 2 ou 3 cellules vivantes, alors elle reste vivante, **sinon elle meurt**.

L'objectif est de simuler, génération après génération, l'état de la grille.

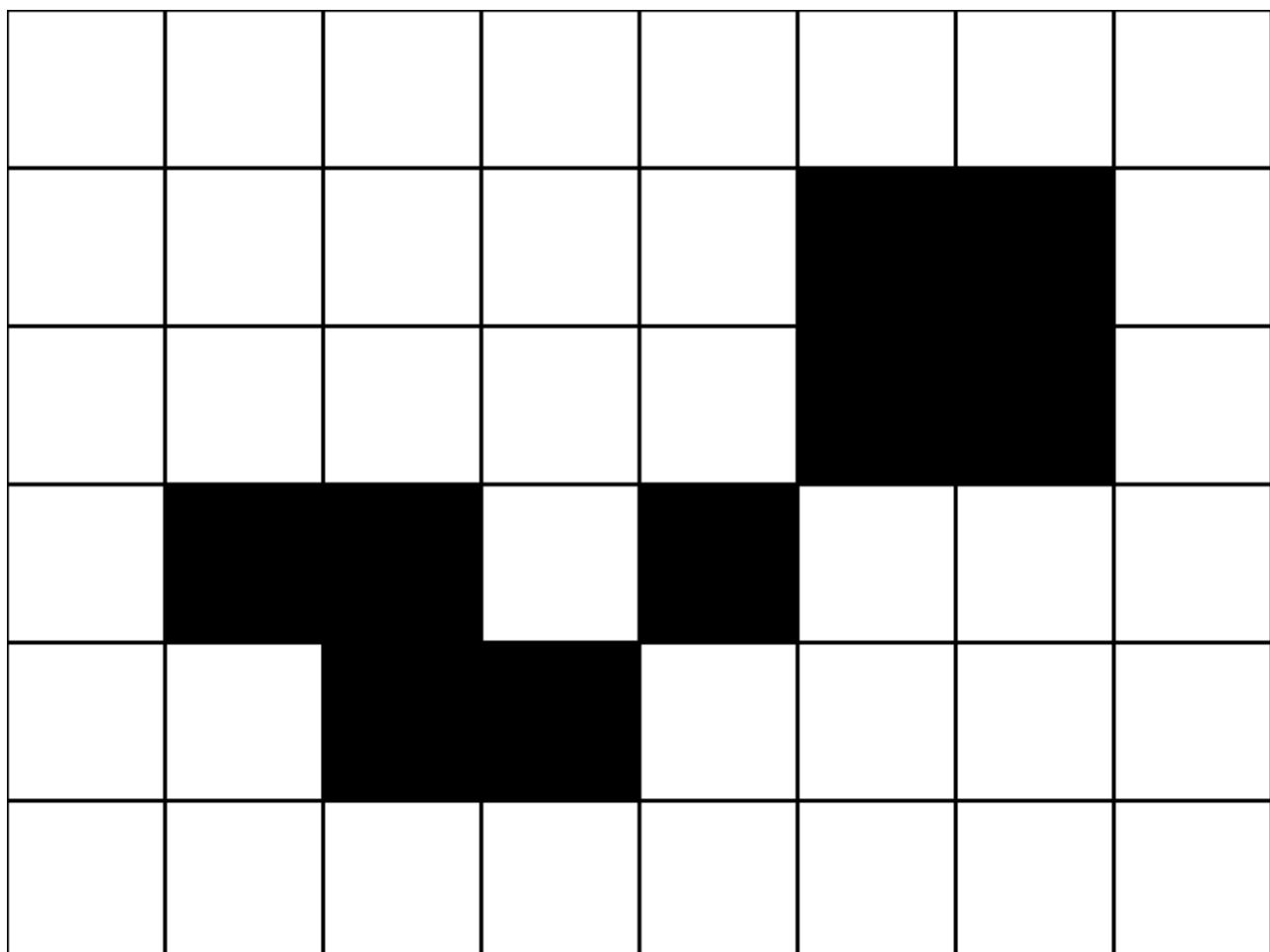
Exemple:

1

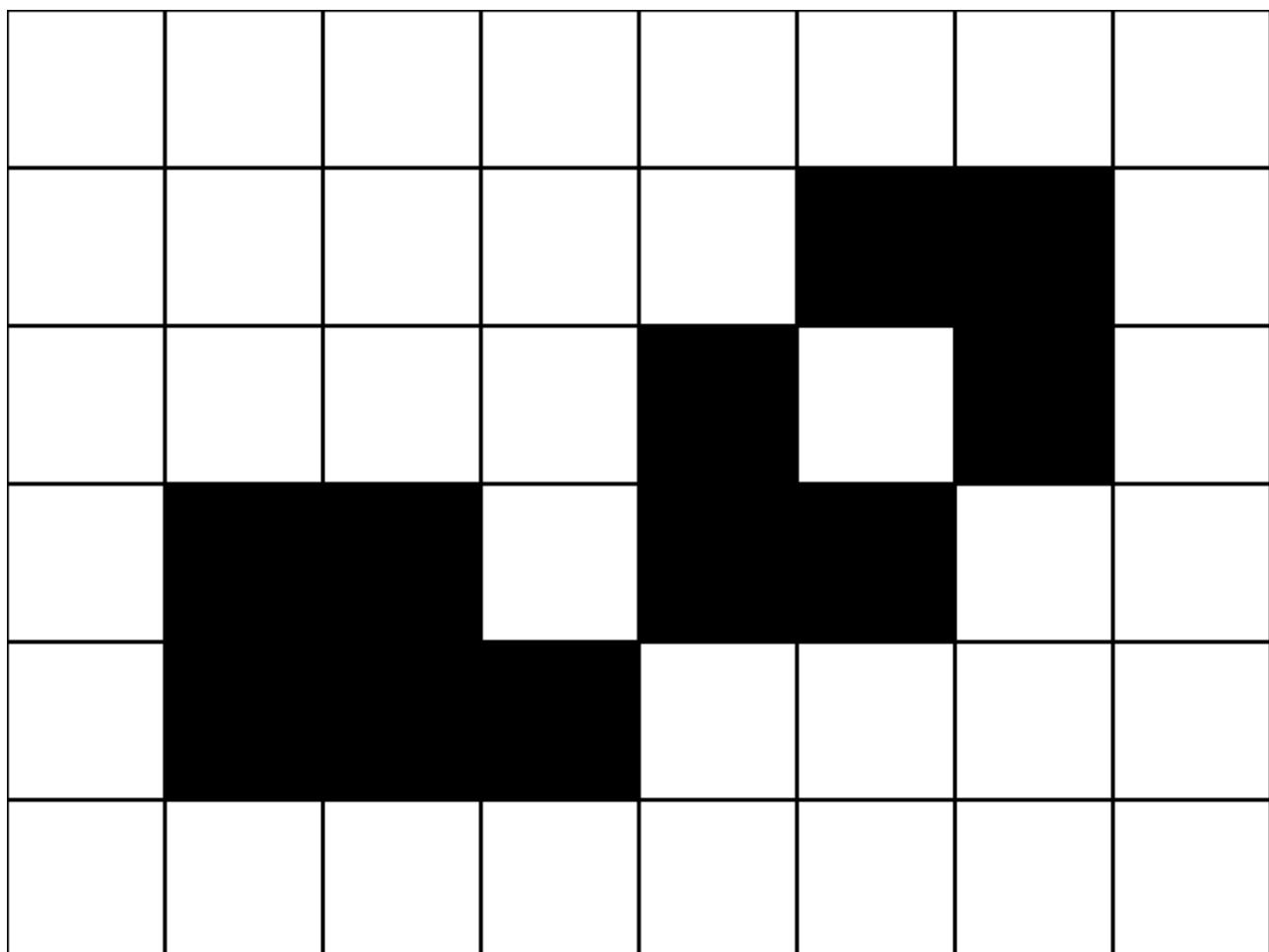
| | | | | | | | | |
|--|--|--|--|--|--|--|--|--|
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |



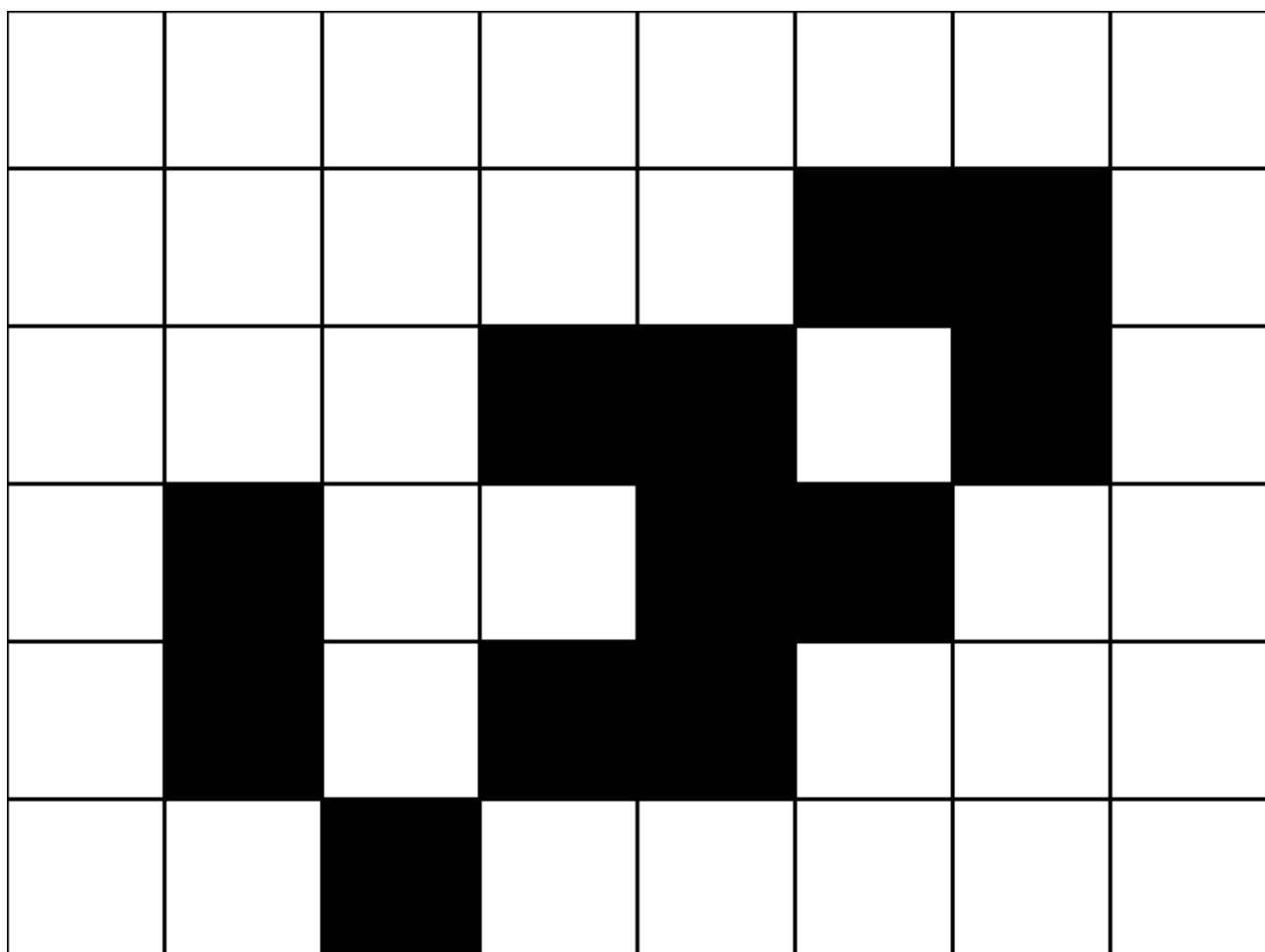
2



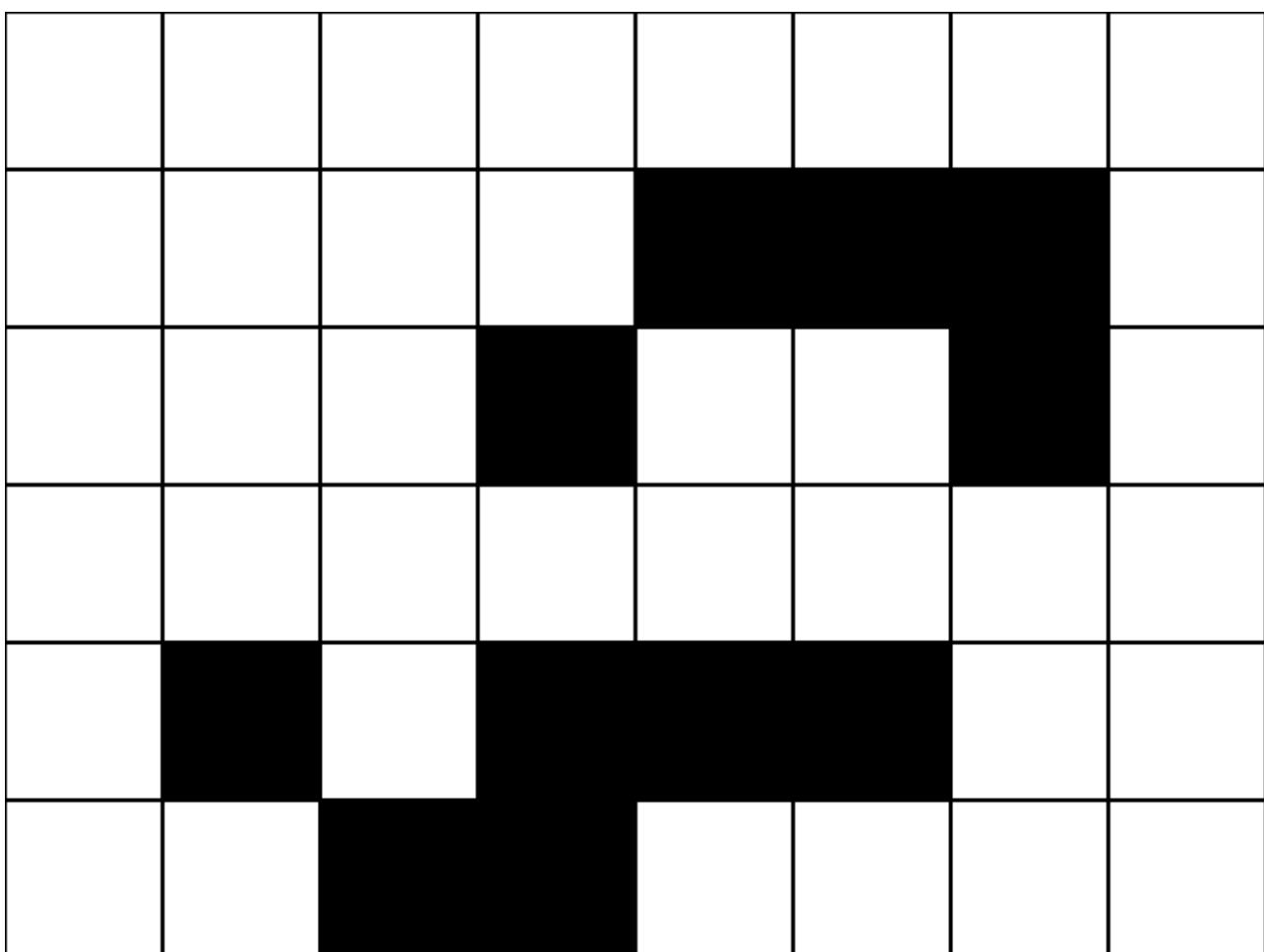
3



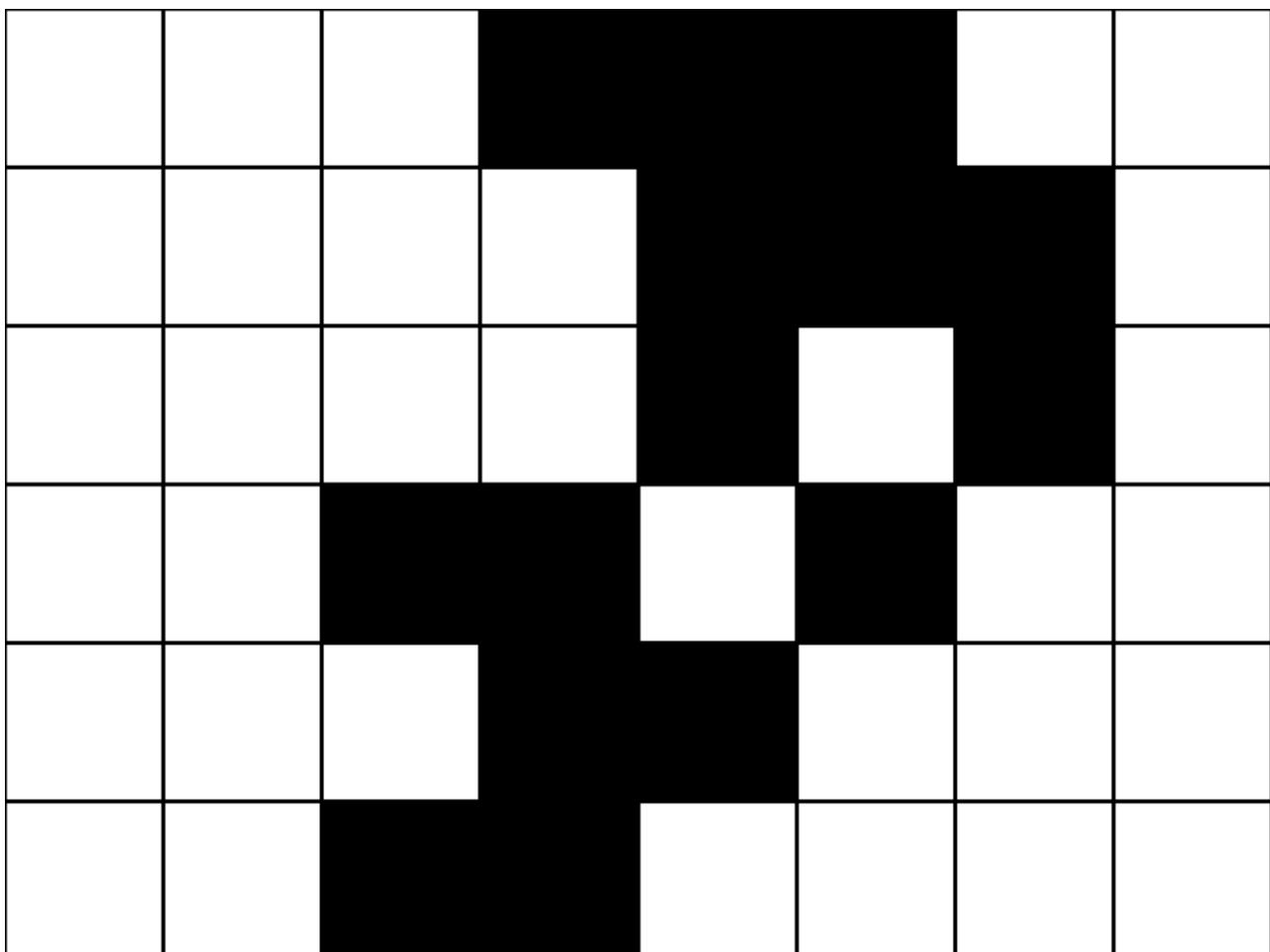
4



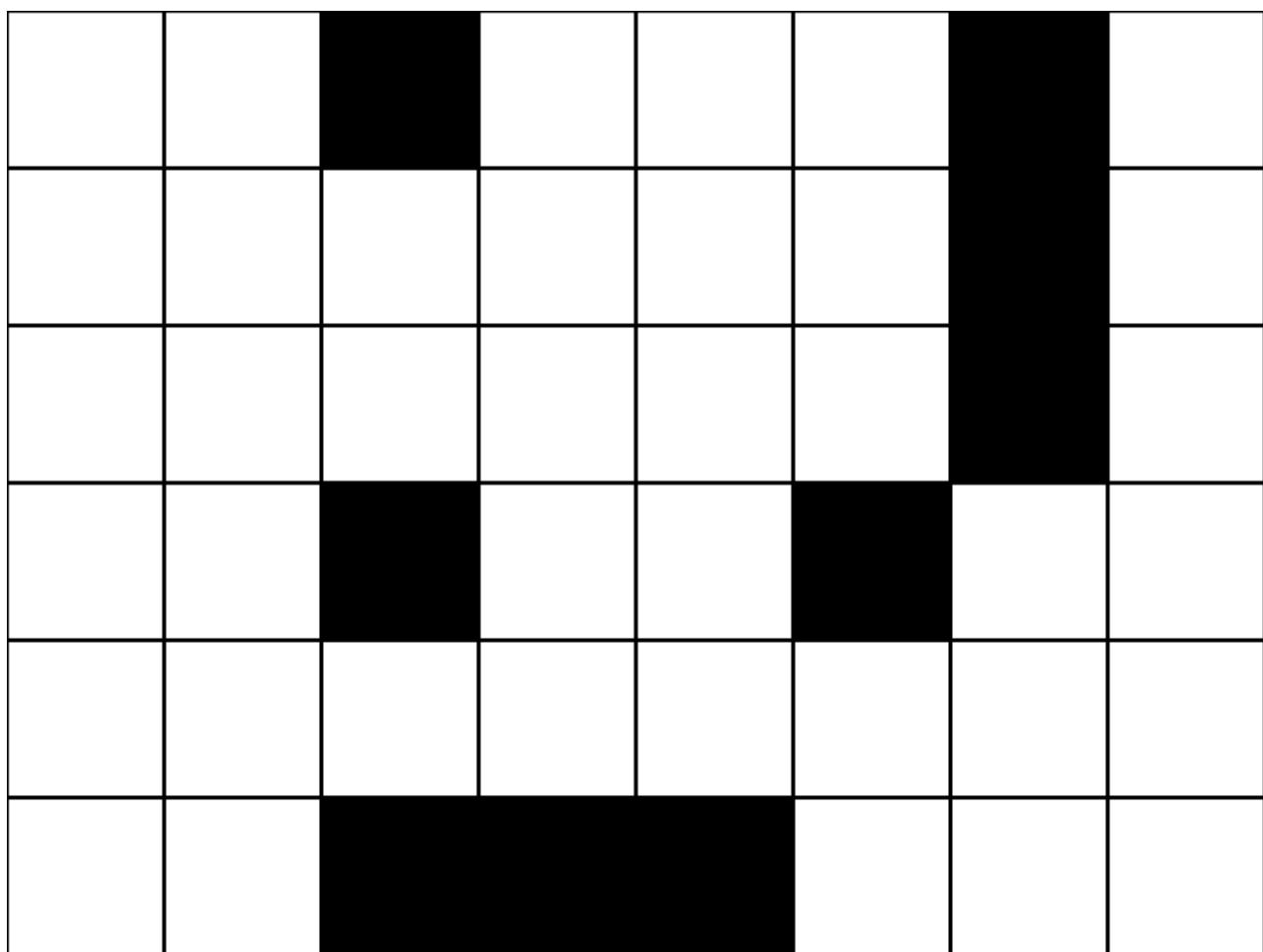
5



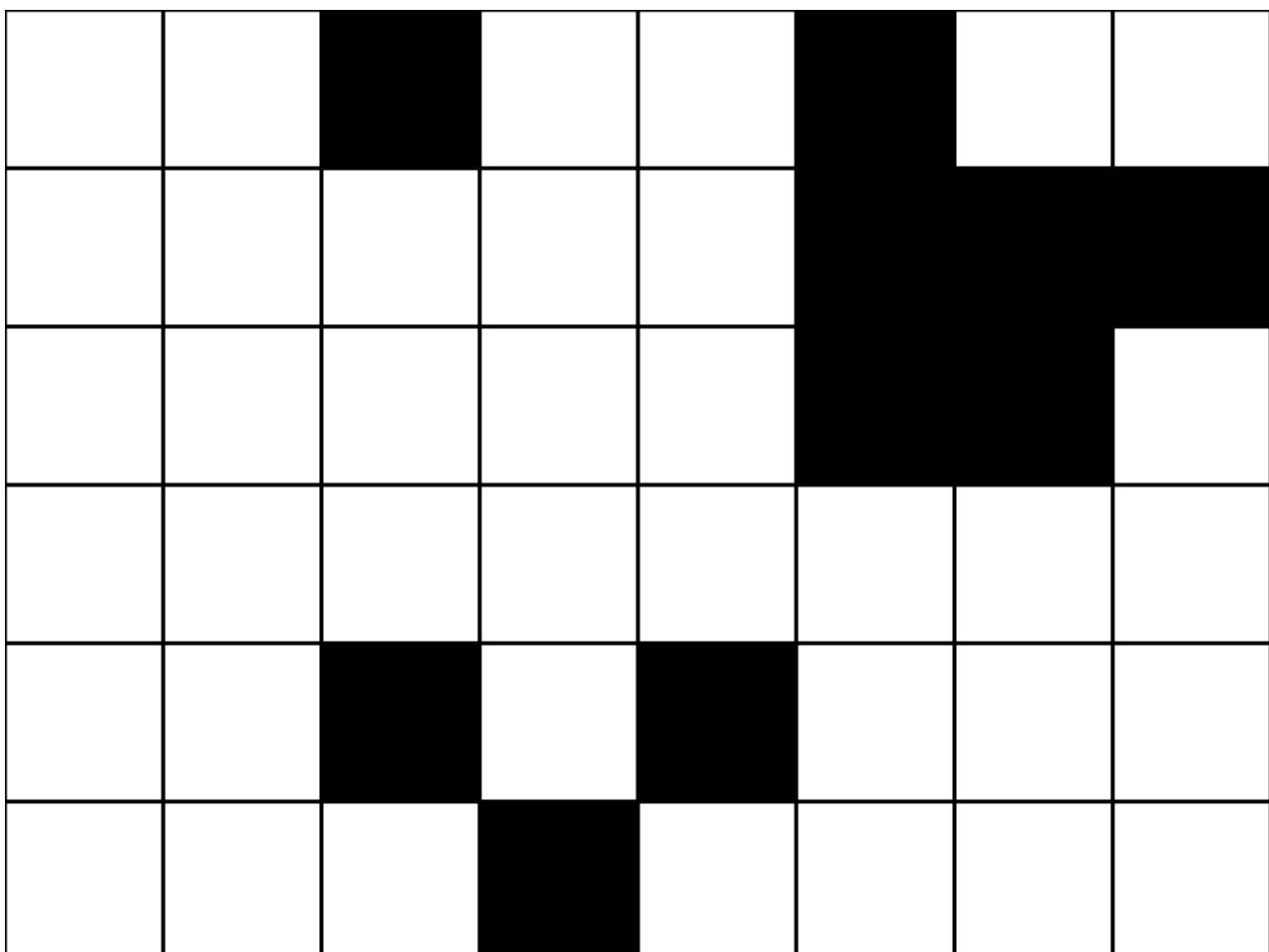
6



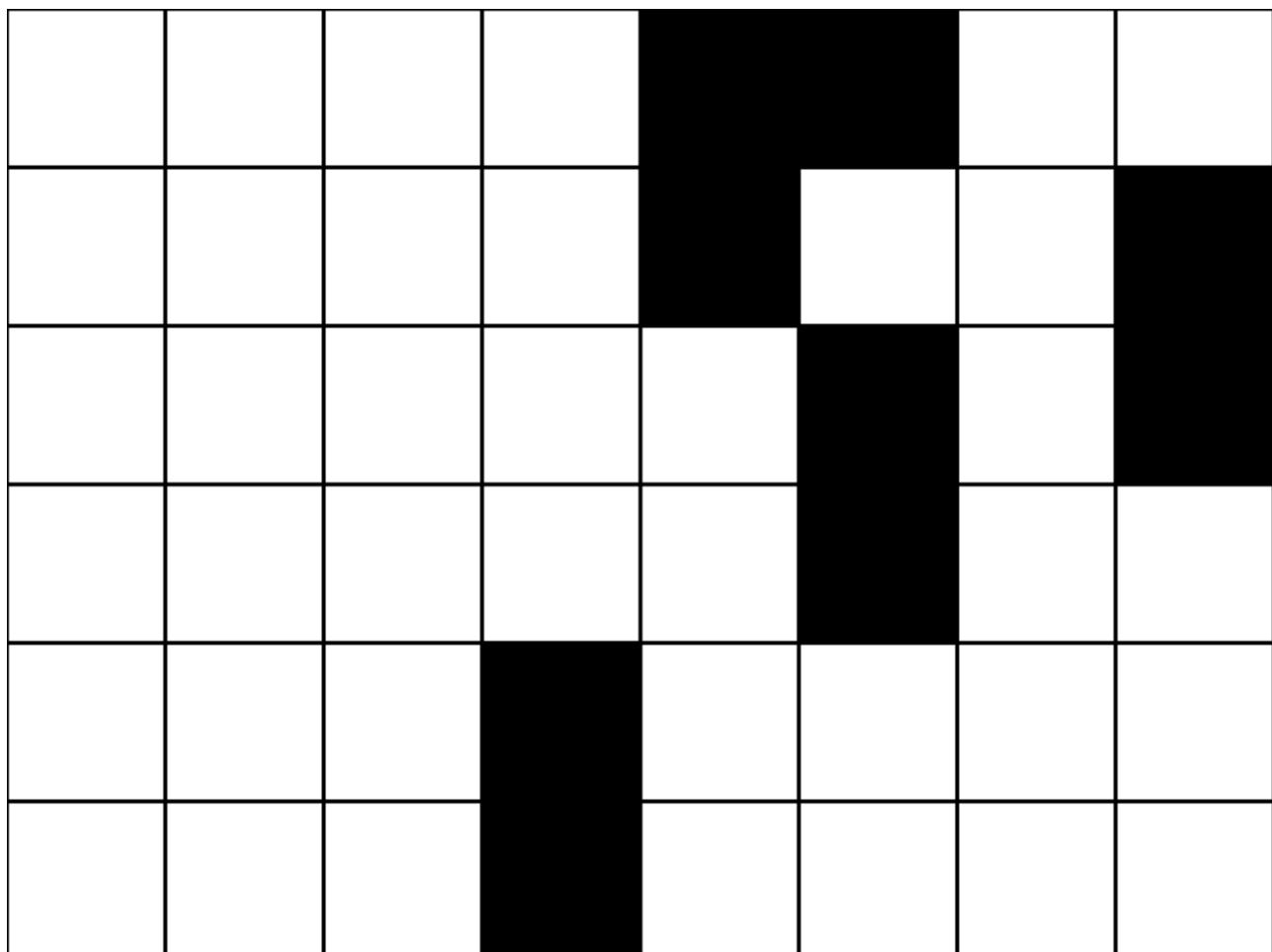
7



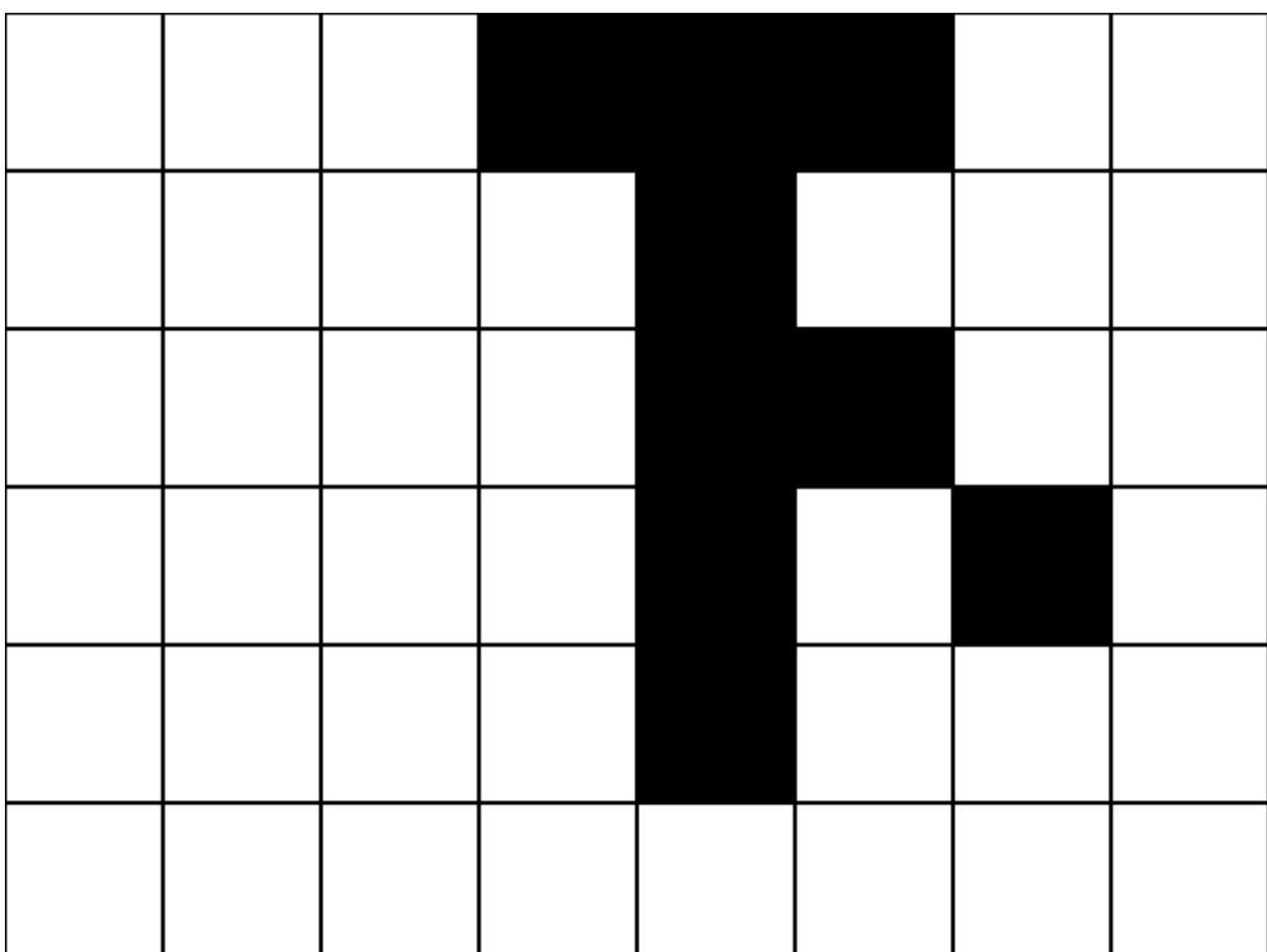
8



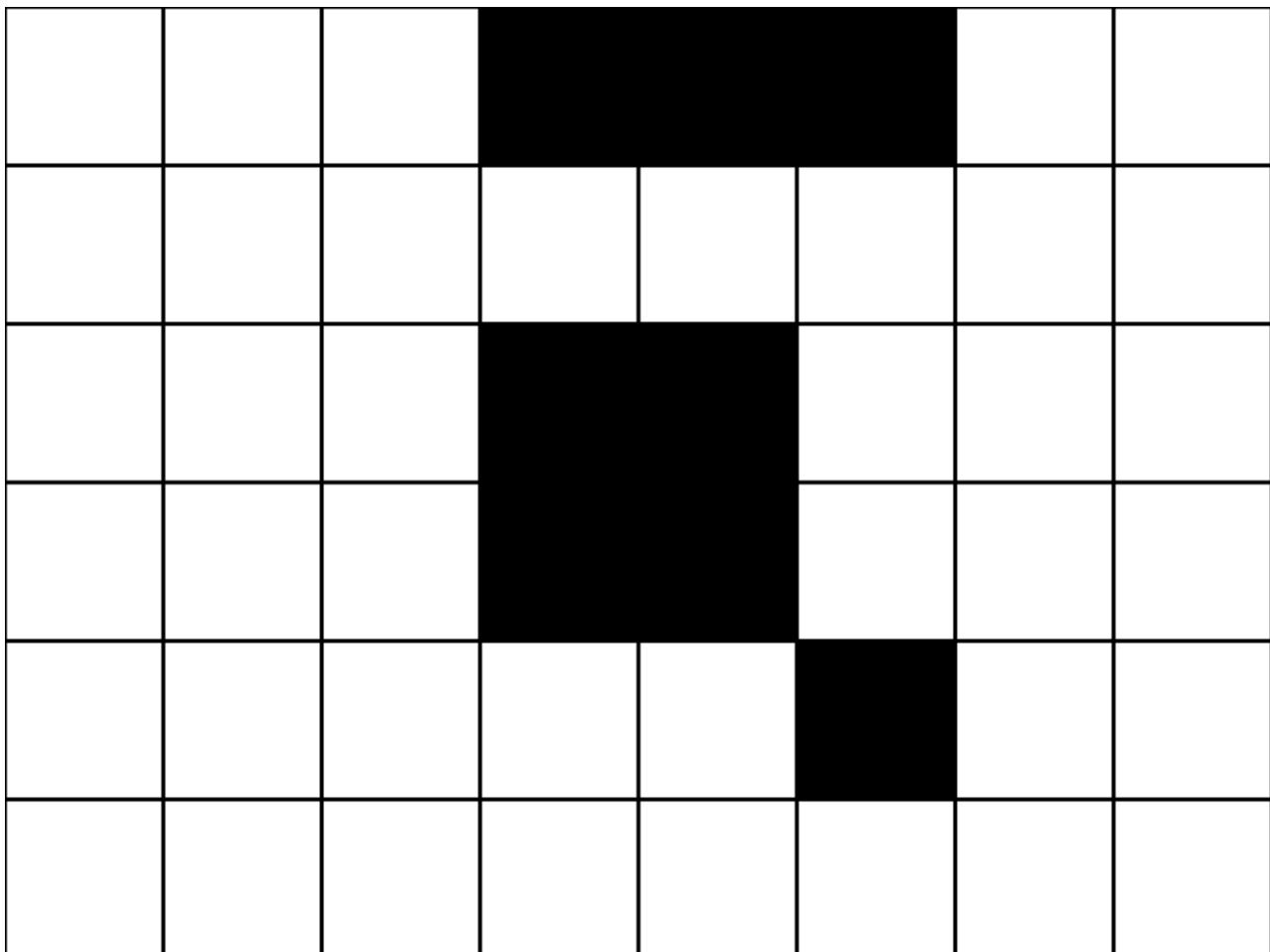
9



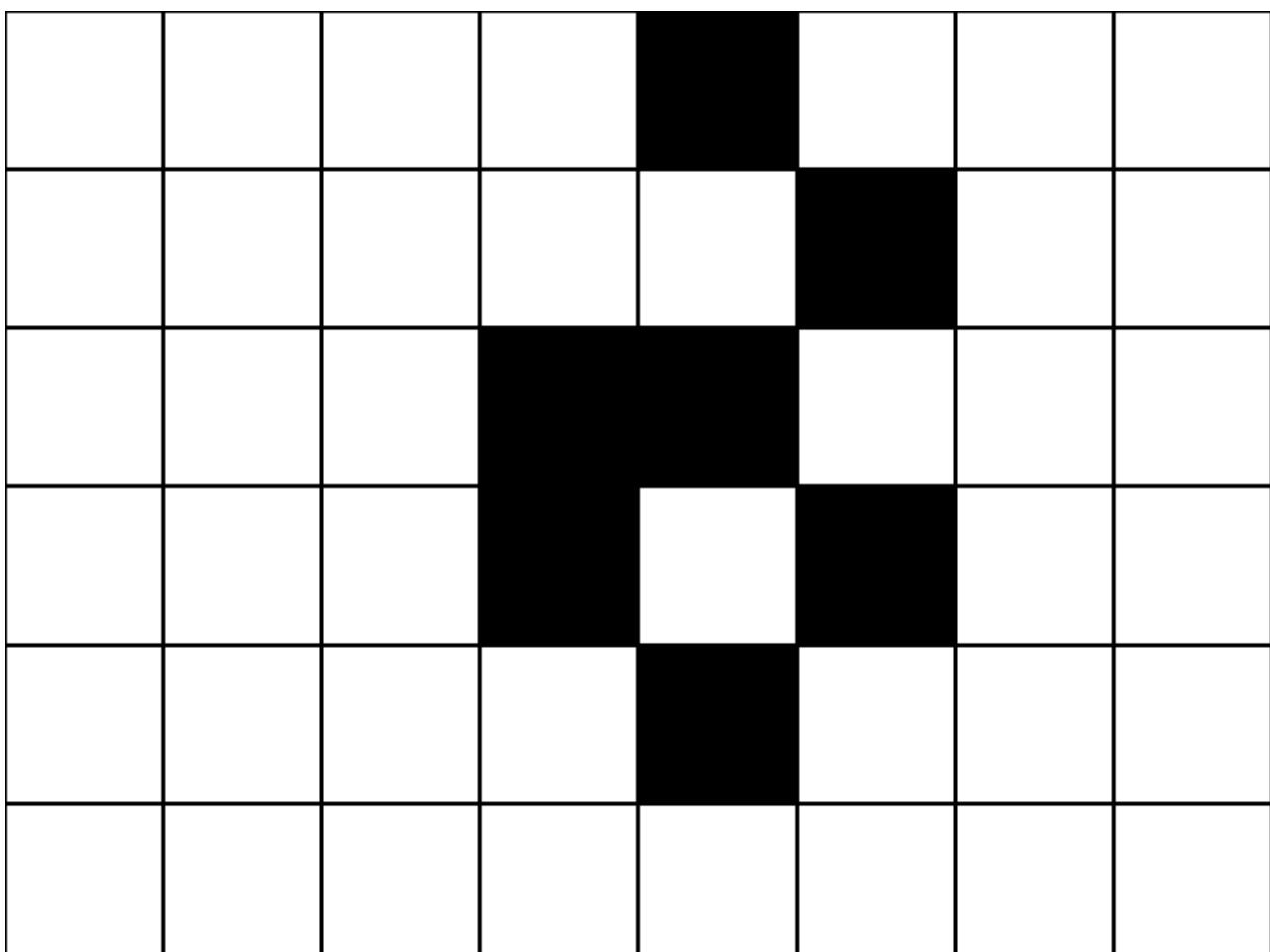
10



11



12



Consignes

- Utiliser le module `pygame` pour animer la grille génération après génération.
- L'état initial de la grille sera choisi aléatoirement.
- Le programme principal devra contenir deux classes: `Grille` et `Cellule`.
- La classe `Grille` contiendra une méthode `actualisation` (ou `update`) qui consistera à actualiser l'état de la grille (c'est à dire de chacune de ses cellules) ainsi qu'à afficher la grille.

La boucle des événements sera donc réduite à (avec par exemple `G` instance de la classe `Grille`):

Script Python

```

1 continuer = True
2 while continuer:
3     for evenement in pygame.event.get():
4         if evenement.type == QUIT:
5             continuer = False
6
7     G.actualisation()
8     pygame.display.flip()
9     pygame.time.delay(100)

```

Proposition de correction

2. Sujet 2: Promenade d'une puce

 Énoncé

Une puce se promène sur une grille dont les cases appelées *cellules* peuvent être blanches ou noires. Au départ, toutes les cellules sont blanches et la puce se trouve au centre de la grille.

La puce peut se déplacer horizontalement ou verticalement sur la grille de la façon suivante:

- si la puce se situe sur une cellule blanche, elle tourne de 90° vers la droite, change la couleur de la case en noir et avance d'une case.
- si la puce se situe sur une cellule noire, elle tourne de 90° vers la gauche, change la couleur de la case en blanc et avance d'une case.

| | | | | | | | |
|--|--|--|--|--|--|--|--|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

 Consignes

- Utiliser le module `pygame` pour animer la grille génération après génération.
- Le programme principal devra contenir deux classes: `Grille` et `Puce`.
- La classe `Grille` contiendra une méthode `actualisation` (ou `update`) qui consistera à actualiser l'état de la grille (c'est à dire de la cellule où la puce est passée) ainsi qu'à afficher la grille.

La boucle des événements sera donc réduite à (avec par exemple `G` instance de la classe `Grille`):

 Script Python

```
1 continuer = True
2 while continuer:
3     for evenement in pygame.event.get():
4         if evenement.type == QUIT:
5             continuer = False
6
7     G.actualisation()
8     pygame.display.flip()
```

 Proposition de correction