

# Olympiades d'informatique – 4<sup>e</sup>

*Sujet – Durée : 2 heures – Travail en équipe*

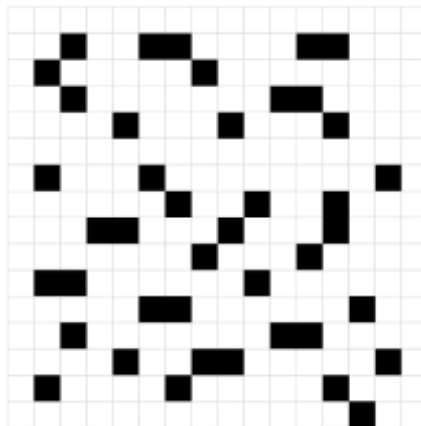
Les deux exercices sont indépendants. Des réponses claires et justifiées sont attendues.

## Problème 1 : le jeu de la vie

Le « jeu de la vie » est un jeu de simulation inventé en 1970 par le mathématicien John Conway.

On part d'une grille composée de cases noires ou blanches représentant des cellules :

- une cellule noire est vivante ;
- une cellule blanche est morte.



À chaque étape, appelée génération, l'état de chaque cellule évolue selon le nombre de cellules voisines vivantes parmi les 8 cases qui l'entourent.

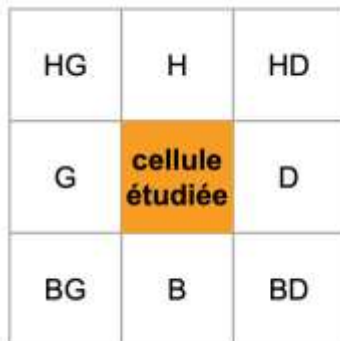
Les règles sont les suivantes :

- une cellule morte ayant exactement 3 voisines vivantes devient vivante ;
- une cellule vivante ayant 2 ou 3 voisines vivantes reste vivante ;
- dans tous les autres cas, la cellule meurt ou reste morte.

On applique ensuite ces règles simultanément à toutes les cellules pour obtenir la génération suivante.

## Schéma du voisinage d'une cellule

Pour appliquer les règles du Jeu de la vie, on regarde une cellule centrale et les 8 cases voisines qui l'entourent et on applique ceci à toutes les cellules.






### Les 8 voisines sont :

haut-gauche, haut, haut-droite, gauche, droite, bas-gauche, bas et bas-droite.

Pour compter les cellules vivantes, on lit l'état de chacune (**0** si elle est morte, **1** si elle est vivante) et on additionne le tout.

### Exemples de situations :

Situation A : 2 voisines vivantes	Situation B : 3 voisines vivantes	Situation C : 4 voisines vivantes
		
On compte 2 cellules noires autour de la cellule centrale.	On compte 3 cellules noires autour de la cellule centrale.	On compte 4 cellules noires autour de la cellule centrale.

### Travail à réaliser

Programmer dans Scratch une simulation du Jeu de la vie (partie A), puis enrichir le programme en ajoutant des actions spéciales afin de modifier l'évolution de la grille (Partie B).

Dans tout le programme, on associe le nombre 0 à une cellule morte et le nombre 1 à une cellule vivante.

Dans ce projet, on travaille sur une grille de 16 colonnes et 16 lignes. Il y a donc 256 cellules au total ( $16 \times 16$ ).

On utilisera le fichier Scratch fourni « **Jeu de la vie.sb3** ». **On commencera par l'enregistrer sous le nom `Jeu de la vie_établissement_NOM1_NOM2_NOM3.sb3`**

Le stockage en mémoire de l'état des cellules de cette grille se fait dans une liste appelée **génération** dans laquelle :

- 0 désigne une cellule morte ;
- 1 désigne une cellule vivante.

La liste **génération** contient donc 256 éléments (les cellules de la grille).

L'état de la cellule de coordonnées (**colonne, ligne**) est stocké dans cette liste à la position :

$$(\text{ligne} - 1) \times 16 + \text{colonne}$$

Pour ranger le contenu des cellules de la grille dans une seule liste, on lit les cellules ligne par ligne, de gauche à droite :

- la cellule de colonne 1, ligne 1 correspond à l'élément 1 de la liste ;
- la cellule de colonne 1, ligne 2 correspond à l'élément 17 de la liste ;
- la cellule de colonne 16, ligne 16 correspond à l'élément 256 de la liste.

L'affichage est réalisé avec des **clones** :

- on crée 256 clones, un par cellule ;
- chaque clone mémorise sa ligne et sa colonne avec les variables **ma\_ligne** et **ma\_colonne** ;
- chaque clone se montre si la cellule est vivante et se cache si elle est morte (**cette partie est déjà codée et n'a pas à être modifiée**).

Enfin, pour passer à la génération suivante, on calcule d'abord tous les nouveaux états dans une nouvelle liste, puis on remplace l'ancienne génération par la nouvelle.

## PARTIE A – Construire le jeu de la vie classique

Défi 1 : compléter le programme Scratch pour obtenir un Jeu de la vie fonctionnel



### Étape 1 – Initialiser une grille aléatoire

Compléter le bloc personnalisé **définir Initialisation de génération**

Quand ce bloc est exécuté, la liste **génération** doit être vidée puis remplie avec 256 valeurs aléatoires, chacune valant 0 ou 1.



Quand ce bloc est exécuté, la liste Grille doit être vidée puis remplie avec 256 valeurs aléatoires (0 ou 1).

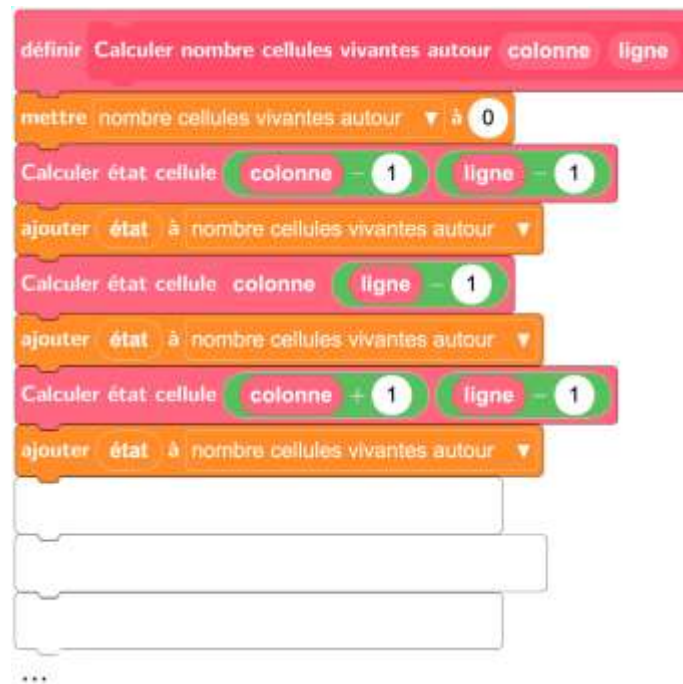
Emplacement dans le fichier Scratch :



### Étape 2 – Compter les voisins vivantes

Compléter le bloc personnalisé **définir Calculer nombre cellules vivantes autour** pour calculer le nombre de cellules vivantes autour d'une case de coordonnées **colonne** et **ligne** parmi les 8 cellules voisines.

Les 5 premiers blocs du bloc personnalisé **définir Calculer nombre cellules vivantes autour** sont donnés.



Aide :

	colonne - 1	colonne	colonne + 1
ligne - 1	$(c - 1, l - 1)$	$(c, l - 1)$	$(c + 1, l - 1)$
ligne	$(c - 1, l)$	<b>cellule centrale</b>	$(c + 1, l)$
ligne + 1	$(c - 1, l + 1)$	$(c, l + 1)$	$(c + 1, l + 1)$

La cellule dite cellule centrale a pour coordonnées (**colonne**, **ligne**).

**Attention** : une cellule située sur un bord n'a pas 8 voisins.

Emplacement dans le fichier Scratch :



### Étape 3 – Calculer une nouvelle génération

Compléter le bloc **définir Calculer nouvelle génération** qui remplit la liste **nouvelle génération** en appliquant les règles du jeu.

Les « trous » à compléter se trouvent aux lignes 4, 6, 11, 12, 14, 15 et sont :

- les nombres dans les deux boucles « répéter »,
- les valeurs **0** ou **1** ajoutées à **nouvelle génération**,
- la valeur de test sur le nombre cellules vivantes autour.

**Bloc à compléter ci-dessous que l'on trouve dans le fichier Scratch.**

```
1 définir Calculer nouvelle génération
2 supprimer tous les éléments de la liste nouvelle génération
3 mettre ligne à 1
4 répéter fois
5   mettre colonne à 1
6   répéter fois
7     Calculer nombre cellules vivantes autour colonne ligne
8     Calculer état cellule colonne ligne
9     si état = 1 alors
10      si nombre cellules vivantes autour = 2 ou nombre cellules vivantes autour = 3 alors
11        ajouter à nouvelle génération
12      sinon
13        ajouter à nouvelle génération
14      sinon
15        si nombre cellules vivantes autour = alors
16          ajouter à nouvelle génération
17        sinon
18          ajouter à nouvelle génération
19      ajouter 1 à colonne
20    ajouter 1 à ligne
21  Copie nouvelle génération dans génération
```

The image shows a Scratch script for calculating a new generation. The script is as follows:

- 1 définir Calculer nouvelle génération
- 2 supprimer tous les éléments de la liste nouvelle génération
- 3 mettre ligne à 1
- 4 répéter fois
- 5 mettre colonne à 1
- 6 répéter fois
- 7 Calculer nombre cellules vivantes autour colonne ligne
- 8 Calculer état cellule colonne ligne
- 9 si état = 1 alors
- 10 si nombre cellules vivantes autour = 2 ou nombre cellules vivantes autour = 3 alors
- 11 ajouter à nouvelle génération
- 12 sinon
- 13 ajouter à nouvelle génération
- 14 sinon
- 15 si nombre cellules vivantes autour = alors
- 16 ajouter à nouvelle génération
- 17 sinon
- 18 ajouter à nouvelle génération
- 19 ajouter 1 à colonne
- 20 ajouter 1 à ligne
- 21 Copie nouvelle génération dans génération

### Rappel des règles :

- Une cellule vivante survit avec **2 ou 3** voisines vivantes.
- Une cellule morte naît avec **exactement 3** voisines vivantes.
- Dans ces deux premiers cas, on ajoute **1** dans la nouvelle génération.
- Dans les autres cas, on ajoute **0** dans la nouvelle génération.

**Important :** on ne doit pas modifier **génération** pendant le calcul, sinon on mélange deux générations.

Emplacement dans le fichier Scratch du début du programme à compléter :



### Étape 4 – Copier la nouvelle génération

Compléter le bloc **définir Copie nouvelle génération dans génération** afin de remplacer l'ancienne liste **génération** par la liste **nouvelle génération**.

On ne modifie pas directement **génération** pendant le calcul, on doit la lire sans la transformer.

Une fois la nouvelle génération calculée, il faut remplacer l'ancienne liste **génération** par la liste **nouvelle génération**.

On écrit donc d'abord dans **nouvelle génération**, puis on effectue la copie à la fin de **nouvelle génération** dans **génération**.

### Méthode attendue en détail :

1. Supprimer d'abord tous les éléments de la liste **génération** ;
2. Parcourir ensuite la liste **nouvelle génération** ;
3. Ajouter chaque élément lu dans **génération**.

Emplacement dans le fichier Scratch :



## Test et observation

Une fois les étapes 1 à 4 terminées :

- activer le mode turbo dans Scratch ;
- lancer la simulation et observer l'évolution.



1. Que se passe-t-il lorsqu'on laisse le programme tourner longtemps ?  
Décrire l'évolution de la grille après plusieurs dizaines de générations.

---

---

---

---

2. Commencer par arrêter votre programme en cliquant sur , puis appuyer sur la touche  du clavier. L'image suivante doit alors apparaître :



Appuyer plusieurs fois sur la barre espace jusqu'à atteindre la génération n°73.  
Dessiner alors ce que vous voyez sur la scène.



## PARTIE B – Extension : actions spéciales

Défi 2 : modifier le programme afin de pouvoir agir sur la grille pendant la simulation

### Action 1 – Bombe

Une bombe placée en (**colonne, ligne**) détruit la cellule (**colonne, ligne**) ainsi que toutes ses voisines (dans un carré de rayon 1).

	colonne -1	colonne	colonne +1
ligne -1	$(c-1, l-1)$	$(c, l-1)$	$(c+1, l-1)$
ligne	$(c-1, l)$	cellule centrale	$(c+1, l)$
ligne +1	$(c-1, l+1)$	$(c, l+1)$	$(c+1, l+1)$

Au maximum, 9 cellules deviennent mortes (valeur 0).

### Expérience 1 – Bombe

1. Programmer un bloc personnalisé Bombe



...

Il est à créer entièrement (pas de script à compléter).

2. Lancer une grille aléatoire et appuyer sur espace 10 fois.
3. Faire une capture d'écran de la génération 10. La sauvegarder dans un fichier.
4. Placer une bombe sur une cellule centrale. On choisit pour cela dans la liste la valeur qui correspond à la cellule (9 ; 9) de la grille.
5. Faire une capture d'écran de la génération 11 puis de la génération 20.
6. Décrire l'effet de la bombe sur l'évolution.

---

---

---

---

## **Bonus (à ne faire que si tout le sujet est fini)**

### **Action 2 – Vaccin**

On souhaite pouvoir vacciner une cellule vivante. Une cellule vaccinée ne peut plus mourir.

Pour simplifier le problème, on va placer une cellule vivante et vaccinée sur une des cases centrales de la grille.

### **Expérience 2 – Vaccin**

1. Programmer un bloc Vacciner qui fixe l'élément de la liste **génération** à 1. (À la fin de l'exécution du programme, la case centrale de la grille doit être à 1 donc son affichage doit rester noir).
2. Modifier le calcul de la nouvelle génération si le bloc Vacciner est activé : si une cellule est vivante et vaccinée, elle reste vivante même si les règles devraient la faire mourir.
3. Relancer une nouvelle grille aléatoire.
4. Appuyer sur espace 10 fois.
5. Faire une capture d'écran de la génération 10.
6. Vacciner une cellule au centre.
7. Faire une capture d'écran de la génération 20 puis de la génération 30.
8. Décrire l'effet du vaccin sur l'évolution.

---

---

---

---

---

## Problème 2 : Création de corail

On établit un ensemble de règles de réécriture qui permet de modéliser la croissance des plantes ou de créer des formes géométriques complexes.

On part d'un mot initial. À chaque étape, on remplace chaque lettre par d'autres selon des règles de croissance précises pour créer des mots de plus en plus longs.

Dans ce problème, chaque lettre correspond ensuite à une action de dessin.

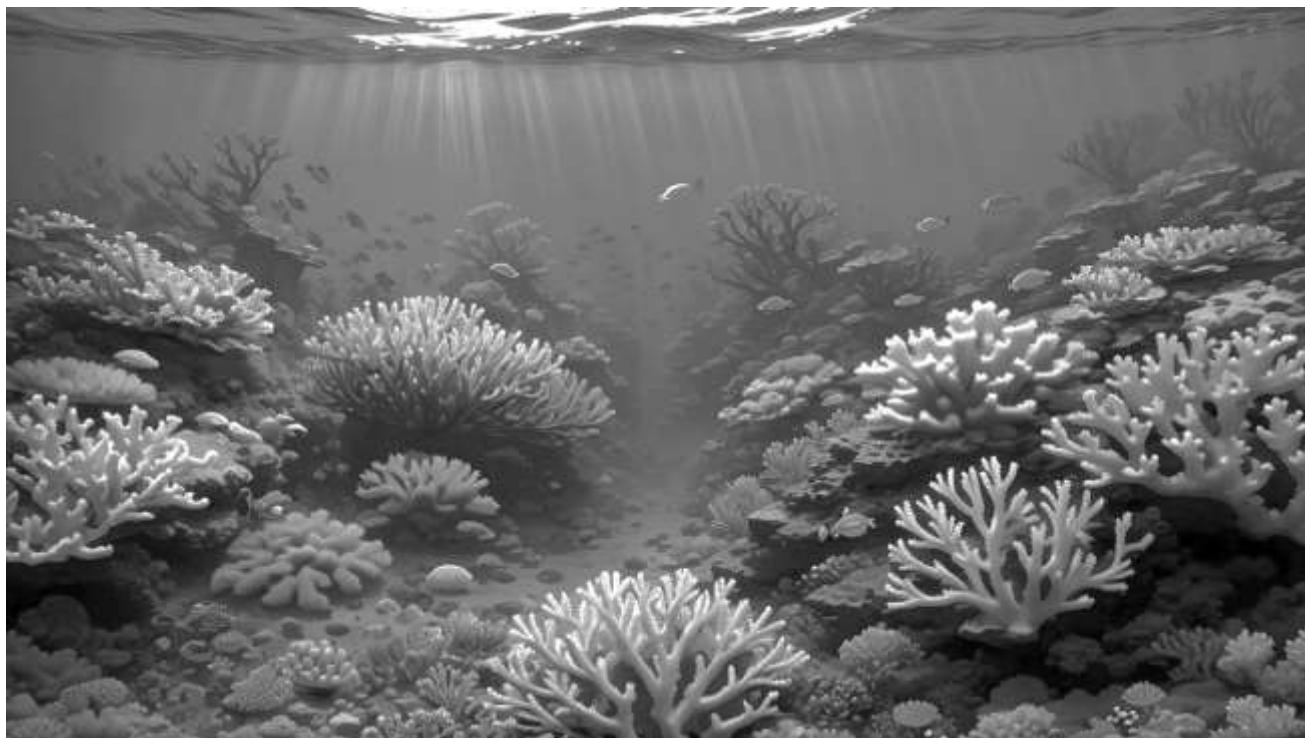


Image d'après Pixabay : <https://pixabay.com/fr/illustrations/ai-g%c3%a9n%c3%a9r%c3%a9-mer-oc%c3%a9an-leau-corail-7992890/>

### PARTIE A : Echauffement

**Objectif** : comprendre un mécanisme de règles de croissance

On a les lettres F et G qui correspondent aux actions suivantes :

<b>F</b>	Tracer un segment de 5 centimètres
<b>G</b>	Pivoter à gauche de 45° sans tracer.

Règles de croissance utilisées :

- La lettre F devient FG : on écrit  $F \rightarrow FG$
- La lettre G devient F : on écrit  $G \rightarrow F$

Le mot initial à l'étape 0 est F.

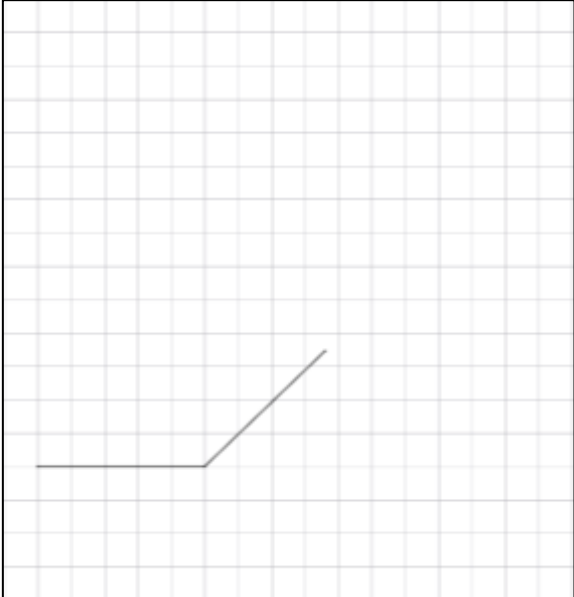
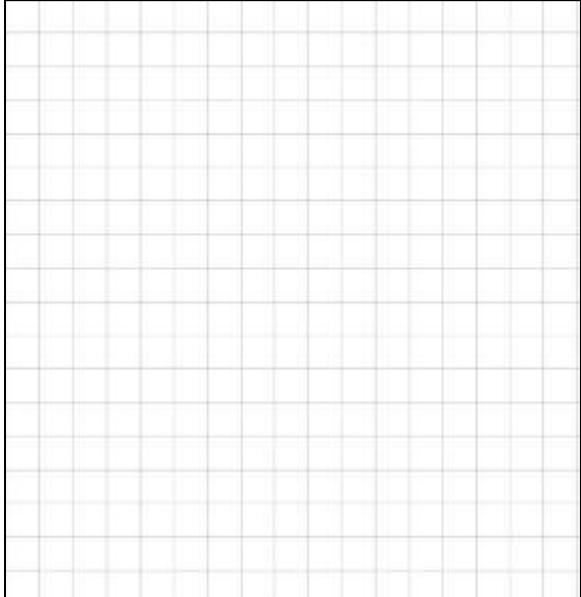
1. Compléter le tableau suivant qui renseigne le mot créé à chaque étape.  
Les étapes 0 et 1 sont déjà complétées.

Étape	0	1	2	3	4
Mot	F	FG			

2. Traduction graphique de ces étapes  
On rappelle les actions suivantes :

<b>F</b>	Tracer un segment de 5 centimètres
<b>G</b>	Pivoter à gauche de 45° sans tracer.

L'étape 2 a été représentée, tracer l'étape 3 sur le quadrillage ci-dessous en prenant 1 carreau pour 1 centimètre.

Dessin de l'étape 2	Dessin de l'étape 3
	

3. Entourer le dessin du mot de l'étape 2 sur le dessin du mot de l'étape 3.

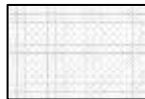
## PARTIE B : Dessin d'un corail

**Objectif** : utiliser d'autres règles pour construire une plante.

On a les lettres F, G, D, R et P qui correspondent aux actions suivantes :

<b>F</b>	Tracer un segment de <b>2 carreaux</b> .
<b>G</b>	Pivoter à gauche de 45° sans tracer.
<b>D</b>	Pivoter à droite de 45° sans tracer.
<b>R</b>	Reculer d'un segment en repassant sur le trait précédent, <b><i>sans changer de direction</i></b> .
<b>P</b>	Dessiner un petit pompon au point où l'on se trouve.
<b>Départ</b>	Point au bas de la feuille à gauche du papier quadrillé fourni, en format paysage, direction vers la droite.

Exemple de format paysage



Règles de croissance utilisées :

- F → F G F P R D D F P R G F
- Les autres lettres (G, D, R et P) restent inchangées.

Le mot initial à l'étape 0 est F.

### Défi 1 : prise en main de ces nouvelles règles de construction

1. Le mot initial à l'étape 0 est F. En utilisant ces règles, combien de lettres F apparaissent à l'étape 1 ?

---

2. Combien de pompons P sont créés à l'étape 1 ?

---

3. D'après vous, pourquoi y a-t-il deux lettres D consécutives au milieu de la règle ?

---

4. À votre avis, quel est le rôle de la lettre R ?

---

## Défi 2. Construction des premiers mots

1. Compléter le tableau suivant :

Étape	0	1	2	3	4
Nombre de F	1				
Pompons déjà présents	0				
Longueur du mot	1				

Méthode possible : observer ce que devient chaque F à l'étape suivante, puis raisonner avec les nombres.

## Défi 3. Prévion

Sans chercher le mot de l'étape 5, prévoir combien il y aura de pompons au total à l'étape 5. Expliquer votre raisonnement.

---

---

---

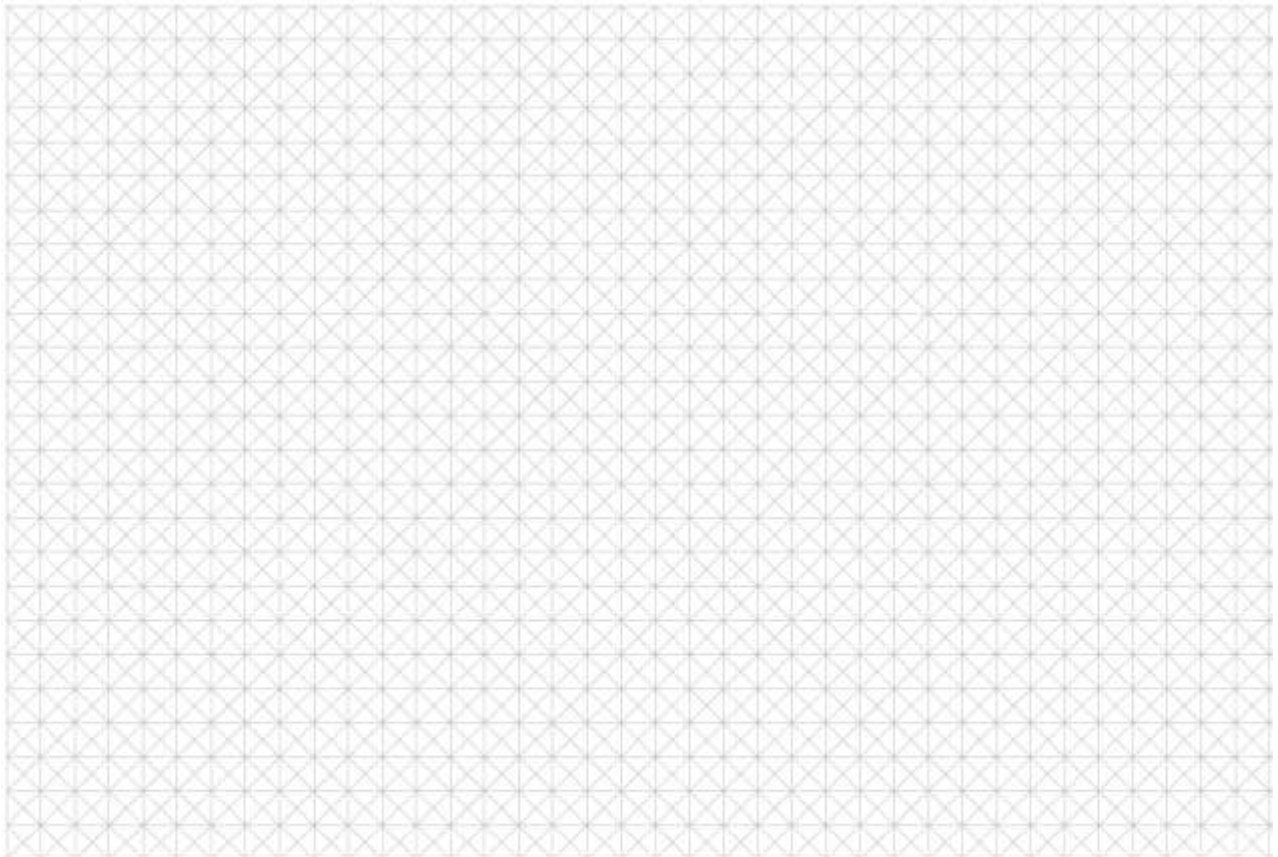
---

---

#### Défi 4. Dessiner la plante progressivement

1. **Consignes** : partir du point indiqué, diriger-vous vers le haut, et prendre 5 carreaux pour un segment F. Les rotations G et D ne tracent rien. La lettre R repasse sur le dernier segment pour revenir au point précédent.  
Faire le dessin de l'étape 2 sur le quadrillage ci-dessous. Le point de départ est en bas à gauche

##### Dessin de l'étape 2



a. À quel endroit les pompons apparaissent-ils ?

---

b. La plante reste-t-elle attachée à sa base ? Pourquoi ?

---

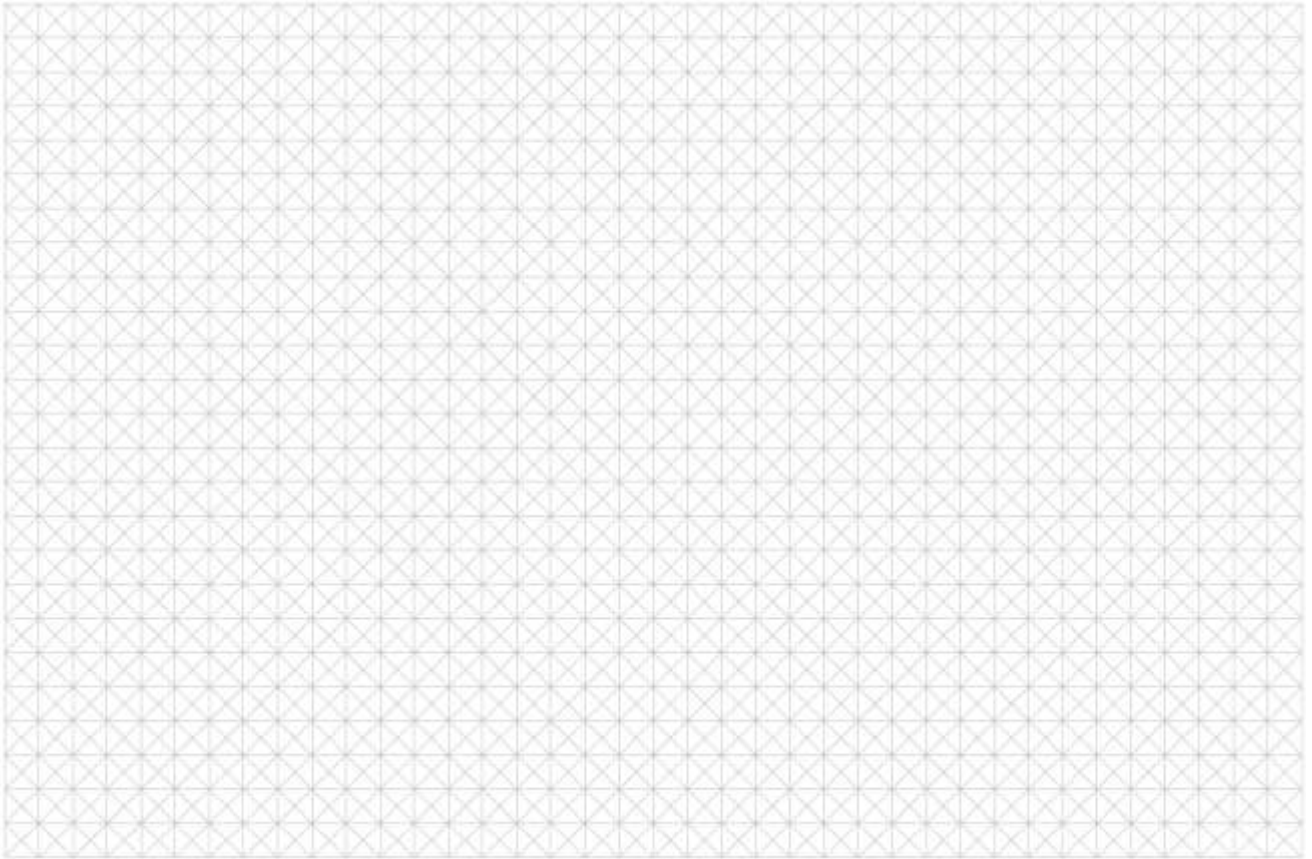
2. Que commence-t-on à voir apparaître : une ligne brisée, un corail, un motif géométrique ?

---

---

### Défi 5. Production de l'étape 3 de la croissance du corail

1. Sur la grille ci-dessous ou sur une feuille quadrillée, réaliser la plante obtenue à l'étape 3. Vous pouvez utiliser une couleur pour la tige, une autre pour les branches et une troisième pour les pompons.



2. Décrire la forme générale de votre corail. Est-ce que certains motifs se répètent ?

---

---

---

---

---



## Partie C. Le botaniste créateur

### Mission 1 : modifier la forme du corail

Proposer une modification des règles pour que le corail ne fasse que monter.

F → \_\_\_\_\_

### Mission 2 : inventer votre corail

À votre tour de modifier la règle pour inventer une nouvelle espèce de corail. Vous devez garder au moins les lettres F, G, D, R et P, mais vous devez ajouter au moins une nouvelle lettre.

Exemples possibles : E = dessiner une épine, C = changer de couleur, X = ne rien tracer mais préparer une branche, etc.

Nouvelles lettres	Description des actions
...	
...	
...	

**Votre nouvelle règle de croissance :**

F → \_\_\_\_\_

Dessiner l'étape 2 de croissance de votre corail et lui donner un nom sur le quadrillage suivant.

