Technologie 5ème	Programmer un objet	
D1.3	***CT 4.1***	*Décrire, en utilisant les outils et langages de descriptions adaptés, la structure et le comportement des objets.*
S 2	***Comment programmer un objet technique ?***	
1/5	Comment créer un organigramme avec le logiciel RobotProg ?	

Lucien souhaite comprendre le fonctionnement de son aspirateur robot. En explorant sur internet il trouve un petit programme développé par Corinne Queme, qui se nomme « RobotProg ». Ce logiciel permet de simuler le comportement de son aspirateur.

Situation(s) problème(s) :

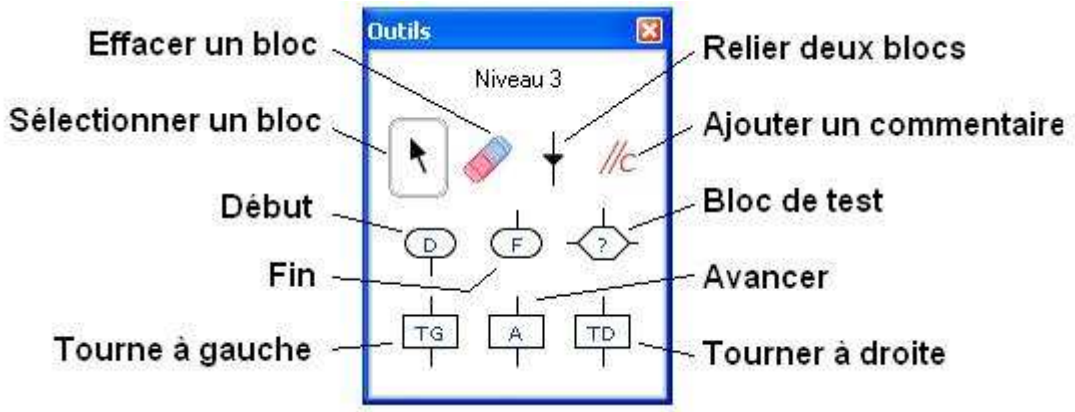
- Qu'est-ce qu'un organigramme de programmation ?
- Quel langage utilise Robotprog ?

Travail

1. Ouvrir le logiciel en double-cliquant sur le logo RobotProg de l'écran d'accueil.
2. Compléter les informations ci-dessous en explorant le logiciel:

Pour programmer votre robot, vous disposez de la fenêtre _____ qui contient tous les blocs nécessaires à la réalisation de vos organigrammes. Pour utiliser un bloc, il suffit de cliquer dessus, puis de cliquer dans la fenêtre ***Programme*** à l'endroit où vous voulez le placer.

Pour l'effacer, il suffit de cliquer sur la _____, puis sur le bloc à effacer.



Pour ouvrir un terrain déjà existant il faut cliquer successivement sur :

_____ => _____ =>_____ =>_____ => choisir le fichier =>_____

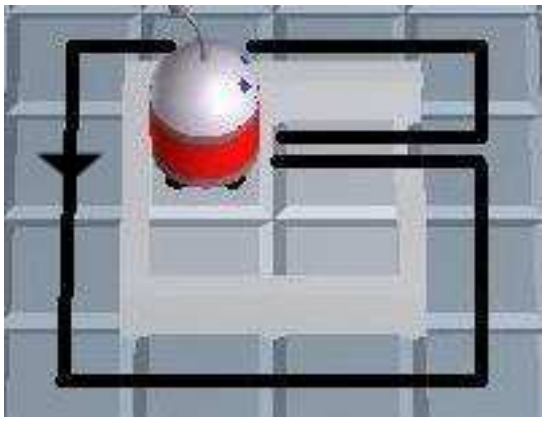
Les blocs début et fin ont une forme : _____ alors que ceux liés aux actions sont : _____.

Si à n'importe quel moment de la séquence tu as un doute, tu peux consulter l'onglet ***Aide*** (en haut à droite). Dedans s'y trouvent, une ***Documentation*** ainsi qu'un ***Résumé du*** _____.

3. Construire un organigramme pour faire avancer le robot d'une case devant lui. Pour le tester, clique sur : ***Exécution*** => ***Initialisation***, puis sur les cases "***INIT***" et "***Lancer l'exécution***" :

Technologie 5ème	Programmer un objet	
D1.3	***CT 4.1***	*Décrire, en utilisant les outils et langages de descriptions adaptés, la structure et le comportement des objets.*
S 2	***Comment programmer un objet technique ?***	
2/5	Comment réaliser un algorithme simple (action par action) ?	

Le robot aspirateur de Lucien peut être piloté par télécommande pour aller nettoyer un endroit de la maison en particulier. C'est très amusant ! Mais comment savoir sur quelle touche appuyer ?

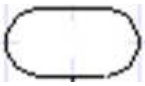


Situation(s) problème(s) :

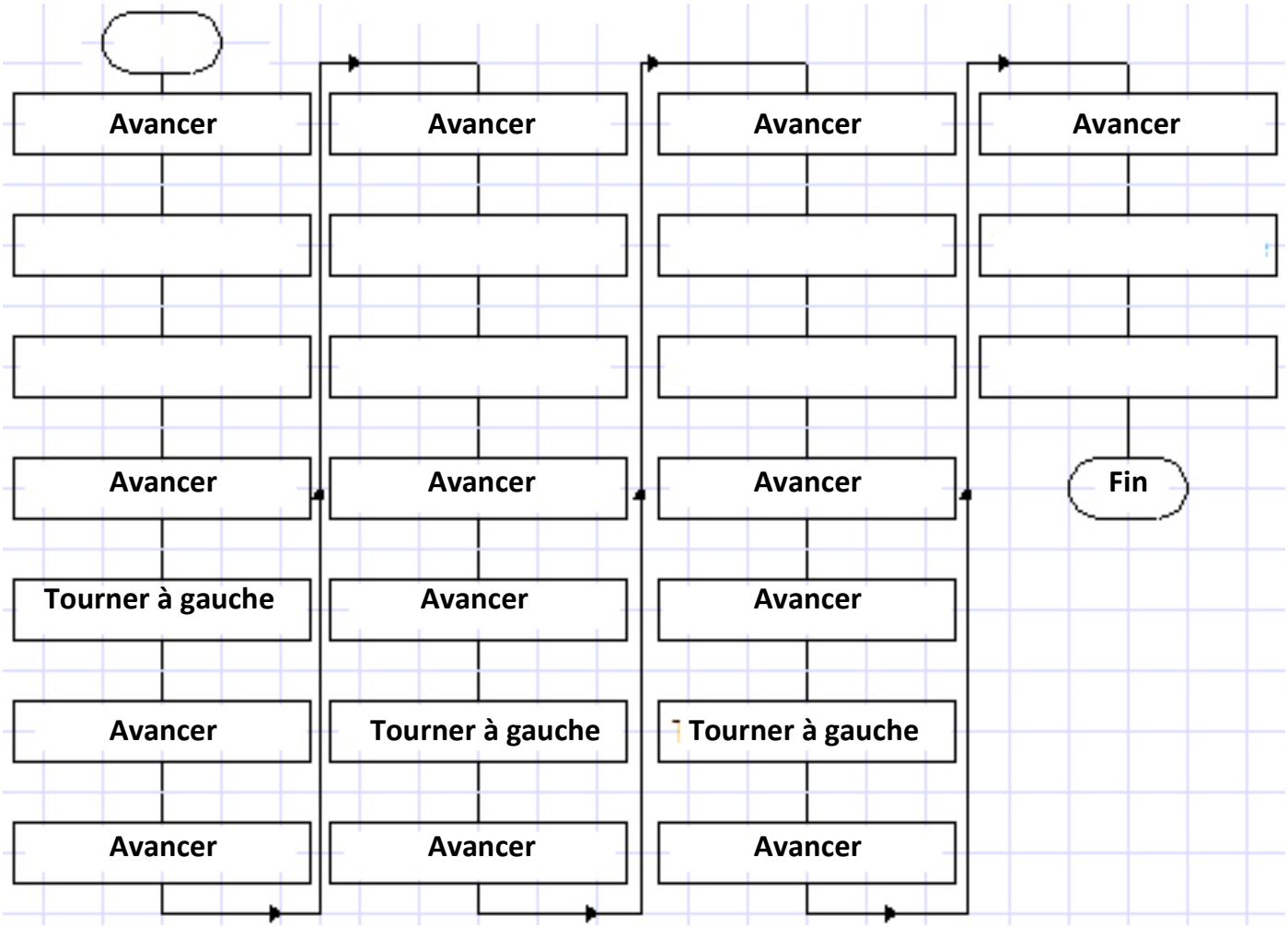
- Comment programmer les déplacements du robot ?

Travail

1. Quel est le seul bloc obligatoire pour chaque algorithme ?

 et il y a généralement à l'opposé une  .

2. Complète l'algorithme pour que le robot fasse le tour de sa maison et revienne dans sa position initiale :



3. Quel est le problème avec cet organigramme ? _____

4. Ouvre le terrain ***Labyrinthe*** du dossier ressources et construis le programme pour aller sur la prise.

=> Bonus : il y a le terrain ***Labyrinthe 2*** pour les plus intrépides d'entre vous…

Technologie 5ème	Programmer un objet	
D1.3	CT 4.1 *Décrire, en utilisant les outils et langages de descriptions adaptés, la structure et le comportement des objets.*	
	CT 4.2 *Appliquer les principes élémentaires de l'algorithmique et du codage à la résolution d'un problème simple.*	
S 2	*Comment programmer un objet technique ?*	
3/5	Comment le robot peut il « réfléchir » ?	

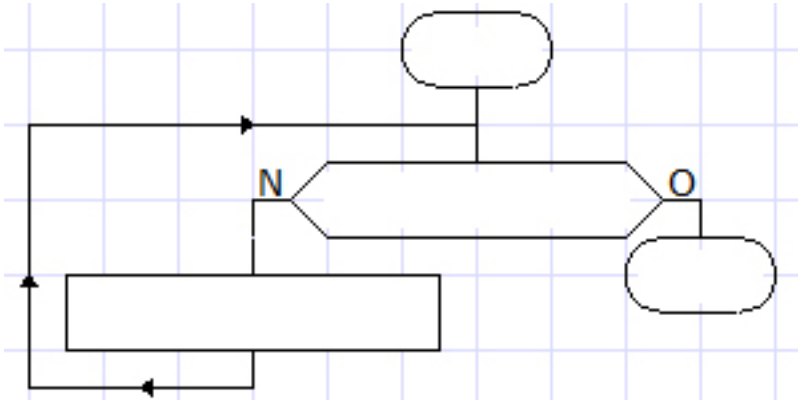
Mais Lucien n'a pas besoin de piloter son aspirateur pour qu'il se déplace. C'est comme s'il était capable de réfléchir et penser lui-même ses déplacements. Lucien va donc essayer d'utiliser des conditions logiques pour rendre son robot autonome.

Situation(s) problème(s) :

- Comment réaliser des tests ?
- Comment simplifier les actions répétitives ?

Travail

1. Dans un organigramme, les conditions logiques sont représentées par un losange. Afin de faciliter la saisie du texte, la forme des blocs de test de RobotProg a été exceptionnellement allongée. Compléter le bloc de test avec le langage du robot (onglet => Aide) afin que le robot vérifie s'il est en face d'un mur.



2. Lorsqu'une action est répétée jusqu'à ce qu'une condition logique soit respectée, on fait une « boucle ». Compléter la boucle de l'organigramme afin que le robot avance jusqu'à ce qu'il soit en face un mur.

3. Ouvrir le logiciel puis le terrain ***terrain vide***. Construire ce programme avant de le tester et de l'enregistrer avec le nom de « ***Avancer au mur en face*** ».

4. Partir de ce programme pour construire l'algorithme qui permet d'aller dans un angle du terrain. Enregistrer ce programme avec le nom de « ***Aller dans un coin***». Afin de ne pas tout ressaisir, tu peux utiliser la méthode du copier/coller. Pour cela, il suffit de :

- Saisir l'outil ***Sélection*** représenté par l'icône :
- Cliquer dans l'angle de la zone des blocs à sélectionner, mais ne relâcher le bouton de la souris qu'une fois arrivé dans l'angle opposé.
- Appuyer simultanément sur les touches ***Ctrl*** + ***C*** du clavier afin de copier la sélection.
- Cliquer dans la zone où coller les blocs qui ont été copiés.
- Appuyer simultanément sur les touches ***Ctrl*** + ***V*** du clavier afin de coller la sélection.

<u>=> Bonus :</u> Les plus rapides peuvent tenter de créer un programme pour que le robot longe le mur à sa droite. Mais attention ! Le robot doit continuer à tourner en rond (enfin en carré…) autour du terrain et non pas se crasher dans le mur d'en face !
Vous pouvez enregistrer le fichier avec le nom « ***Suivre le mur de droite*** »

Extraits du langage du robot :

MurEnFace	Fonction logique retournant la valeur vrai si le robot est en face d'un mur
MurADroite	Fonction logique retournant la valeur vrai s'il y a un mur à droite du robot
MurAGauche	Fonction logique retournant la valeur vrai s'il y a un mur à gauche du robot

Technologie 5ème	Programmer un objet	
D1.3	***CT 4.2***	*Appliquer les principes élémentaires de l'algorithmique et du codage à la résolution d'un problème simple.*
S 2	***Comment programmer un objet technique ?***	
4/5	Comment faire si on ne connait pas la position du robot ?	

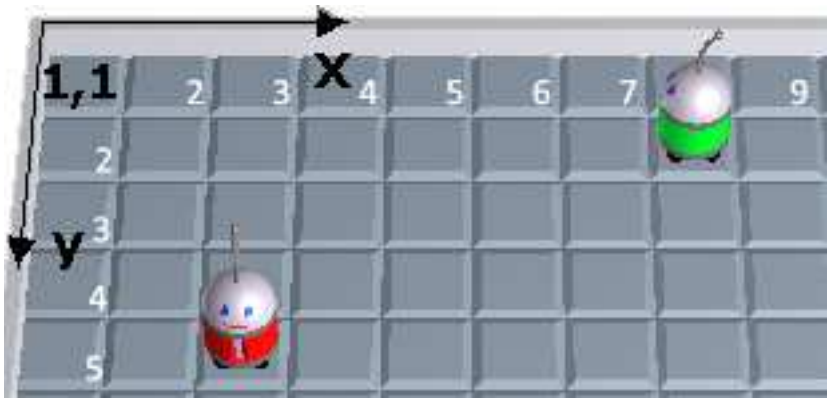
Maintenant Lucien se demande comment son robot peut retrouver et revenir directement à sa base, peu importe le trajet aléatoire qu'il a effectué. Il aimerait donc essayer de programmer son petit robot pour qu'il aille automatiquement sur une case.

Situation(s) problème(s) :

- Comment interpréter une position ? une direction ?
- Qu'est ce qu'une variable ?
- Comment atteindre une colonne ? Une ligne ? Une case ?

Explications:

Même si nous ne connaissons pas la position de départ du robot, le logiciel enregistre ses coordonnées avec deux variables. Ce sont des informations identifiées par un nom spécifique, afin que nous puissions y faire référence dans nos algorithmes sans avoir besoin d'en connaitre la valeur.

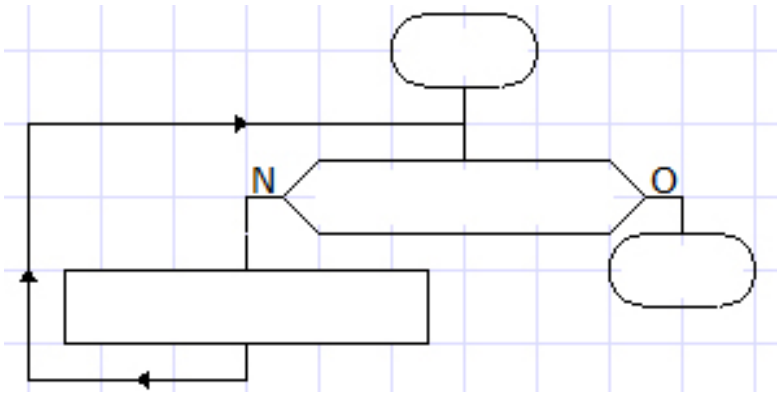


Ainsi, dans la ***Documentation***, partie ***Langage du robot***, on nous explique que le terrain est composé de 2 axes nommés ***x*** (colonnes) et ***y*** (lignes). Les coordonnées du robot sont enregistrées sous les variables ***xRobot*** et ***yRobot*** avec pour valeurs la position du robot sur ces axes. Le coin en haut à gauche est donc la case (1;1) et se prononce : case (***x***;***y***). Pour s'en rappeler facilement, le ***x*** est avant le ***y*** dans l'ordre alphabétique et quand tu écris sur une feuille tu commences toujours en haut, de gauche à droite puis de haut en bas…

Travail

1. Donne les valeurs pour le robot rouge : ***xRobot*** = ____ et ***yRobot*** = ____ .
2. Donne les valeurs pour le robot vert : ***xRobot*** = ____ et ***yRobot*** = ____ .

3. Complète l'algorithme suivant afin de faire avancer le robot jusqu'à la colonne 9.

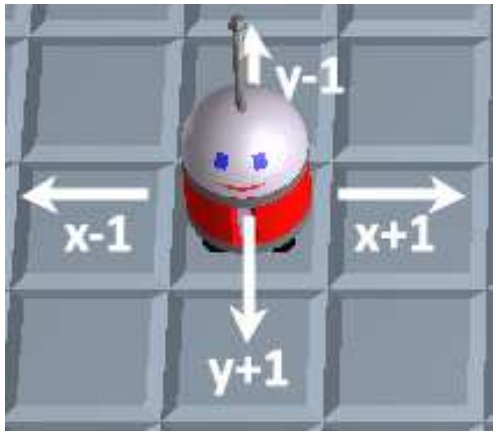


4. Construis le programme « ***Avancer colonne 9 »*** et ouvre le terrain nommé ***croisement*** afin de le tester correctement. Tu vas voir apparaitre une restriction t'expliquant qu'il faut être au niveau 2. Pour le débloquer cliques sur : ***Configuration*** => ***Niveau*** => ***6*** => ***Ok.*** De nouveaux outils devraient apparaitre au passage. Ils nous servirons plus tard...

5. Tu peux placer ton robot n'importe où sur le terrain. Le fichier fonctionne toujours sauf dans 2 cas :
   - S'il est dans les 8 premières colonnes, il faut qu'il soit : _____
   - S'il est sur la colonne 10, il faut qu'il soit : _____

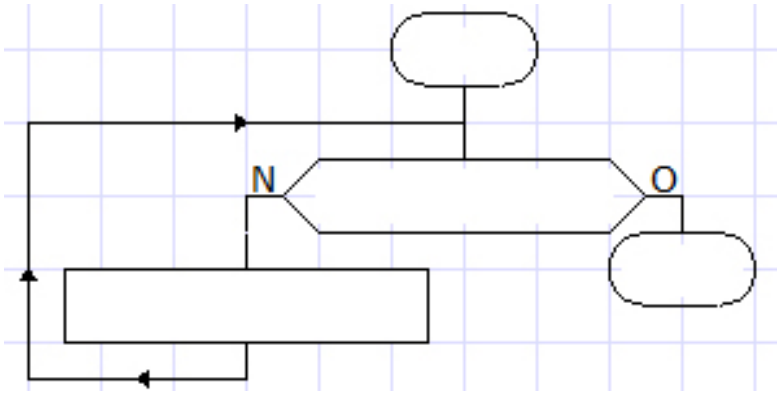
Explications:

Comme nous devons aussi préciser l'orientation du robot, il existe 2 variables pour donner la direction du robot : ***dxRobot*** (direction du robot sur l'axe des x) et ***dyRobot*** (direction du robot sur l'axe des y). Elles sont également expliquées dans la ***Documentation***, partie ***Langage du robot.*** Mais pour résumer simplement, il suffit de regarder quelle variable est modifiée si le robot avançait d'une case dans la direction qu'il regarde. Par défaut leur valeur est de 0. Si c'est horizontal (gauche ou droite) alors c'est ***dxRobot***, et si c'est vertical (haut ou bas) alors c'est ***dyRobot.***



→ : ***dxRobot*** = 1
← : ***dxRobot*** = -1
↓ : ***dyRobot*** = 1
↑ : ***dyRobot*** = -1

Travail



6. Créer un programme qui permette de faire pivoter le robot jusqu'à ce qu'il soit dirigé vers la droite. Enregistre-le sous le nom de « ***Diriger vers la droite*** » et recopie içi ← son organigramme.

Attention à ne pas confondre *Diriger vers la droite* et *Tourner à droite*! Diriger, c'est orienter par rapport au terrain, alors que tourner, c'est juste faire ¼ de tour avec le robot…

Bonus : les plus rapides peuvent essayer d'assembler les 2 programmes afin de diriger le robot vers la droite avant de le faire avancer jusqu'à la colonne 9, et recopier l'organigramme içi ↓:

Technologie 5ème	Programmer un objet	
D1.3	CS 5.7	*Analyser le comportement attendu d'un système réel et décomposer le problème posé en sous-problèmes afin de structurer un programme de commande.*
S 2	***Comment programmer un objet technique ?***	
5/5	Comment rassembler toutes ces informations?	

Lucien a compris qu'il pouvait faire de nombreux programmes. Mais cela devient très complexe de les assembler avec un organigramme très imposant et compliqué à lire. Or, il a vu dans les nouveaux outils qui sont apparus avec le niveau 6, qu'il existe des sous-programmes. Cela semble pratique…

Situation(s) problème(s) :

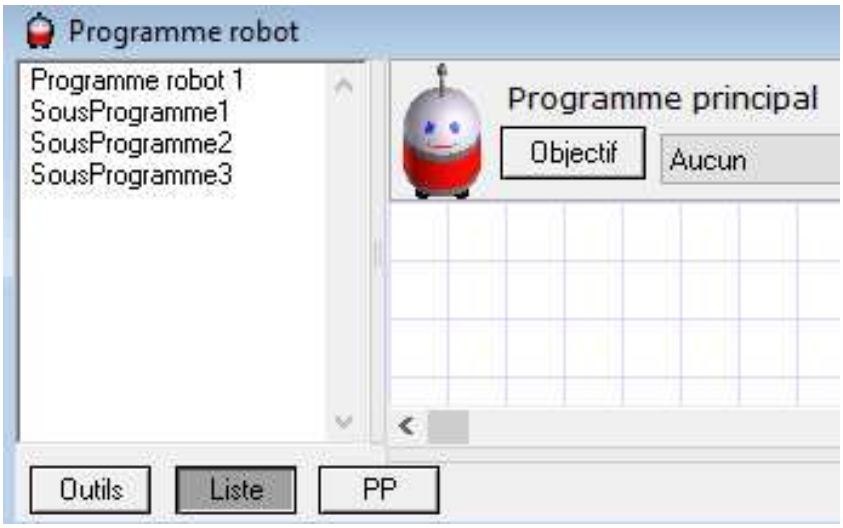
- Comment utiliser des sous-programmes pour assembler plusieurs programmes en un?

Explications:

L'outil ***Appel de sous-programme*** permet d'insérer un bloc qui représente à lui seul un algorithme complet. Ce bloc est de forme rectangulaire, comme un bloc d'action, mais possède une 2ème bordure verticale de chaque coté.

Il renvoie à un sous programme qu'il faudra insérer puis impérativement renommer avec le même nom. Pour cela, il faut cliquer sur : ***Programmation*** => ***Nouveau sous-programme.***

Afin de naviguer facilement entre le programme principal et les sous-programmes, vous avez la possibilité d'afficher la liste de vos organigrammes en cliquant simplement sur ***Liste***. De même en cas de doute vous pouvez cliquer sur ***PP*** pour revenir au ***P***rogramme ***P***rincipal.



Travail

1. Grâce aux sous-programmes et la méthode du copier/coller, utiliser les algorithmes précédemment sauvegardés pour construire un programme nommé « ***colonne 9 complet*** ». Il doit permettre de se rendre jusqu'à la colonne 9, et ce, peu importe la position et la direction de départ du robot (schéma 1).
2. Modifier le programme afin qu'il serve à aller à la ligne 4 et l'enregistrer sous « ***ligne 4 complet*** ».
3. Assembler les deux programmes en un seul pour se rendre automatiquement case (9;4) (schéma 2).

Schéma 1 :

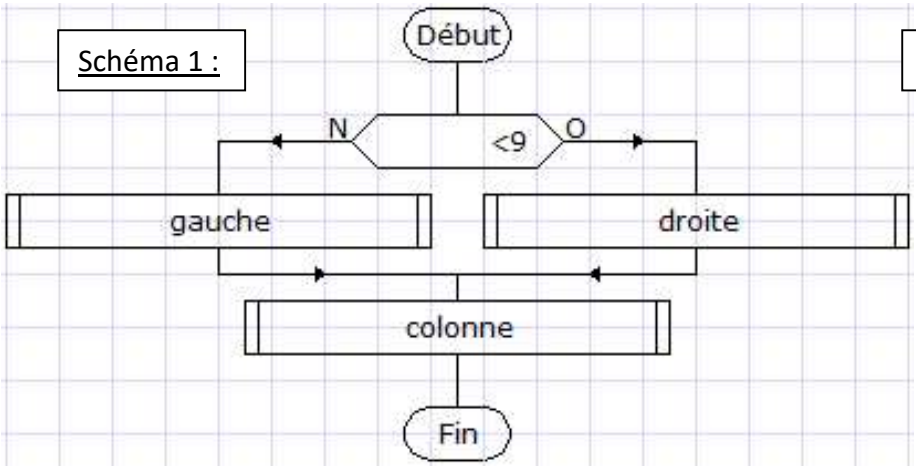
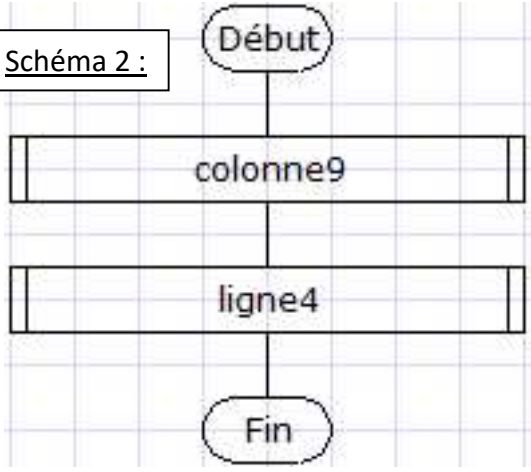


Schéma 2 :



Bonus : Vous pouvez essayer de trouver l'algorithme permettant de résoudre les 2 labyrinthes…