CTP SDA2 - 9 Nov 2018 (2h)

Documents autorisés (vos fichiers personnels de TP)

Consigne 1 :commencez par créer un répertoire nommé ctpsda2nov18<NOM>

Consigne 2 : en fin d'épreuve, vous déposerez une archive de ce répertoire sur moodle.

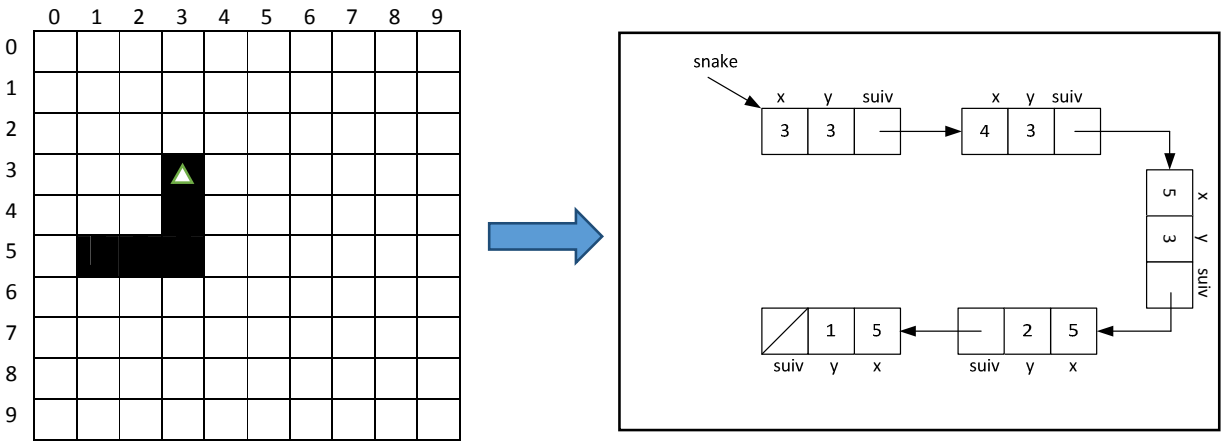
Consigne 3 : veuillez rendre des codes qui compilent !

et si possible, conviviaux

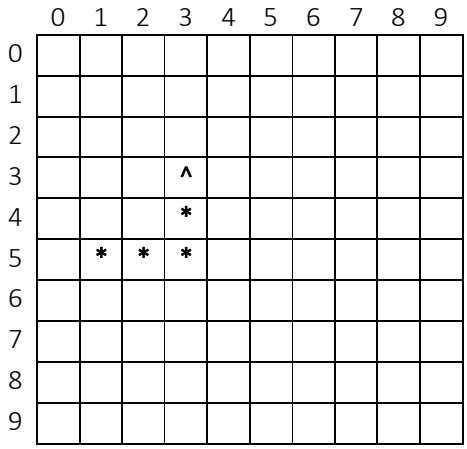
Exercice 1 (ex1.c) : SNAKE 10pts

Dans le cadre du développement du célèbre jeu de serpent *snake*, l'équipe de développement a décidé d'implémenter le *snake* en utilisant une liste chainée. Chaque noeud de cette liste correspond à un maillon du *snake* et contient les coordonnées x et y de ce maillon. Le *snake* évalue dans une grille de 10 * 10 cases et peut posséder de 2 à 9 maillons.

La figure suivante montre l'état du *snake* a un moment donne ainsi la liste chainée correspondante :

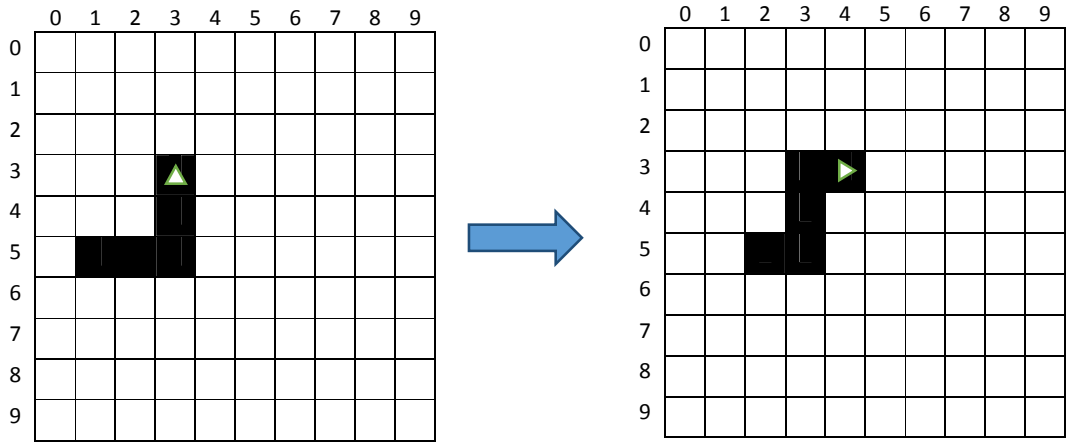


1. Ecrire une structure de données qui permet de représenter le *snake*.
2. Ecrire une fonction **insererQueue** qui permet d'ajouter un nœud en queue afin de créer un snake.
3. Ecrire une fonction **afficheSnake** qui permet d'afficher le *snake* sur la grille (matrice) de la manière suivante :



- **^ pour la tête du *snake* et * pour les autres nœuds.**

4. Le snake se déplace d'une seule case à chaque étape vers une direction qui peut être : **Nord, Sud, Est ou Ouest**. Le mouvement du snake se fait simplement en ajoutant un nœud en tête et une suppression en queue. Les coordonnées de la nouvelle tête du *snake* sont calculées en fonction de la direction voulue.



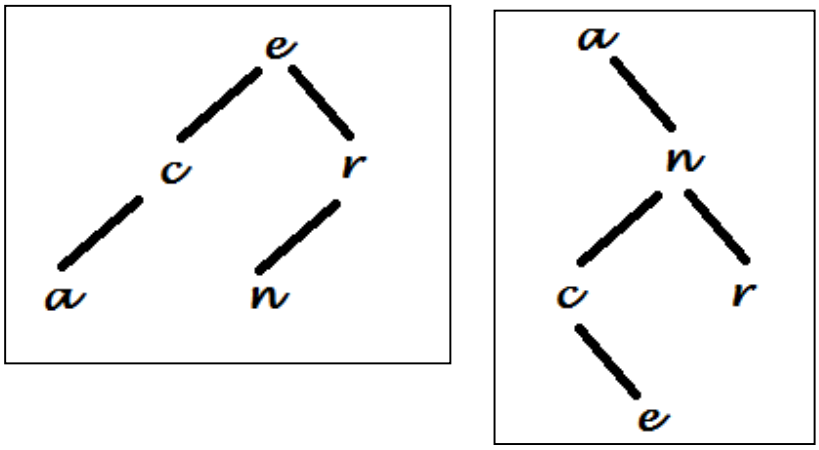
- Ecrire une fonction **mouvement** qui permet de faire avancer d'une seule case sur la grille un *snake* donnée en paramètre vers une direction donnée (**Nord, Sud, Est ou Ouest**). Il faut prendre en compte le fait que le *snake* peut sortir d'une des extrémités de la grille et apparaitre à l'extrémité opposée.

5. Ecrire un programme qui permet de tester les fonctions précédentes.

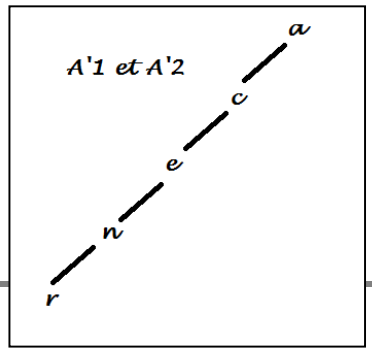
Exercice 2 (ex2.c) : Anagrammes 10pts

L'idée de cet exercice est de comparer deux chaines de caractères afin de savoir si nous sommes en présence d'anagrammes.
Exemple : **ecran** et **ancre** (on ne gèrera que les lettres minuscules sans accents, sans cédille, etc ...)

1. **écrire un programme demandant de saisir deux chaines qui seront rangées lettre par lettre, dans respectivement deux arbres A1 et A2 (arbres de char).**



2. **Ecrire une fonction int anagrammes(T_ABR a1,T_ABR a2) qui comparera les deux arbres afin de renvoyer 1 si les chaines sont des anagrammes.**
   <u>Deux approches sont possibles (au choix) :</u>
   - **Première approche : on comparera les deux arbres tels qu'ils sont mémorisés (en RAM, comme ci-dessus) et on vérifiera que chaque lettre de l'un appartient à l'autre. On s'assurera aussi que les deux arbres ont le même nombre de noeuds.**
   - **Deuxième approche : durant un parcours en infixé de chaque arbre , on crée deux autres arbres A'1 et A'2 qui seront alors identiques. Il reste alors à les comparer noeud à noeud.**



!! : pour toute autre approche, vous expliquerez votre raisonnement et votre démarche pour aboutir au résultat souhaité