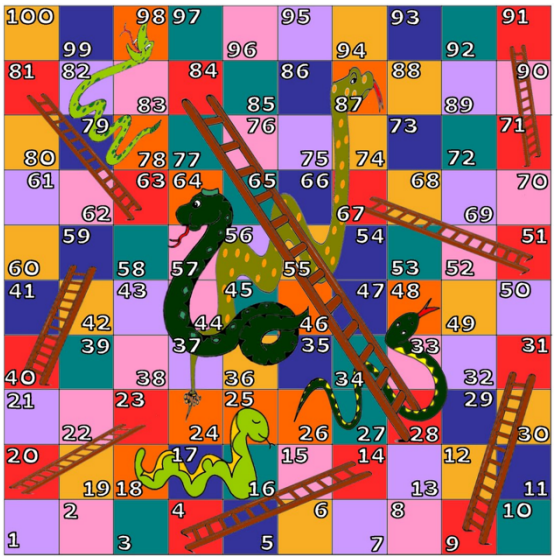
Mini-projet : jeu échelles et serpents (rendu pour le vendredi 18 février)

L'objectif de ce projet est de programmer en version informatique le jeu bien connu des enfants : serpents et échelles. Ce jeu est une variante du jeu de l'oie. La version programmée sera pour un seul joueur, l'objectif étant de réussir en un minimum de coups.



La règle est la suivante :

- Le joueur lance deux dés à 6 faces ;
- Il avance d'un nombre de cases correspondant au total des deux dés :
  - Si la case d'arrivée est une **échelle**, le joueur, le joueur **avance** jusqu'à la case correspondant à l'autre extrémité de l'échelle ;
  - Si la case d'arrivée est la tête d'un **serpent**, le joueur **recule** jusqu'à la case correspondant à la queue du serpent ;
  - Si le total des dés ajoutés au numéro de la case dépasse 100, le joueur **recule** d'un nombre de case égal à la différence (ex. le joueur est à la case 97, le joueur obtient 4 et 5, le joueur avance de 3 pour atteindre la case 100 puis recule de 6 pour atteindre la case 94).
- Le jeu se **termine** quand la case 100 est atteinte avec un nombre exact de cases à traverser.

A. Travail attendu

Pour programmer du jeu serpents et échelles, vous disposez d'un squelette (**`serpents.c`**) de programme à compléter consistant à :

- Définir la structure de données (type **`CaseJeu`**) pour représenter une case du jeu et la grille du jeu sachant que :
  - Chaque case est numérotée par un nombre entier strictement positif ;
  - La grille du jeu doit être représentée par une liste chaînée de cases ;
  - Une échelle et un serpent seront représentés par un pointeur vers la case destination.
- Générer la grille du jeu (fonction **`lire_fichier_grille`** à compléter) à partir d'un fichier texte (**`serpents-echelles.txt`**) sachant que :
  - Le fichier doit être utilisé tel quel sans modifications ;
  - Une ligne représente une case du jeu. Chaque ligne est composée de deux nombres entiers : le numéro d'une case et le numéro d'une case destination s'il y a une échelle ou un serpent (0 sinon) ;
  - Les cases dans le fichier ne sont pas ordonnées.
- Gérer le jeu (fonction **`jouer_jeu`**) en fonction des entrées utilisateur conformément à la règle décrite ci-dessus sachant que :
  - Avant chaque lancé de dés, le jeu demande à l'utilisateur s'il continue ou arrête complètement le jeu ;
  - Des messages indiqueront l'avancement du jeu à chaque lancés. Voici un exemple d'interaction avec l'utilisateur en mode texte :

```
Vous etes en case 1
Continuer (pour arreter saisir 0) ? 1
Vous lancez les dés : 4 et 4
Vous arrivez en case 9
Un échelle : vous avancez en case 31
[...]
Continuer (pour arreter saisir 0) ? 1
Vous lancez les dés : 3 et 3
Lancés de dés trop élevé : vous reculez en case 95
```

- Libérer la mémoire allouée lors de la génération de la grille (fonction **`supprimer_grille`** à compléter).

Outre les fonctions mentionnées ci-dessus, vous aurez besoin de définir des fonctions supplémentaires. A vous de déterminer les fonctions nécessaires.

B. Conseils

Ce projet est un travail de synthèse portant les notions vues dans les modules S1106 et S1103 (sauf les arbres) :

- S'appuyer sur les TP du module S1106 ainsi que les TP vu dans ce module S1103 ;
- Décomposer en plusieurs fonctions pour simplifier votre programme ;
- Reprendre et adapter les algorithmes vus en cours et TP (ex. pour insérer une case en fonction de son numéro, penser que la case peut déjà exister) ;
- Ne pas chercher de nécessairement un algorithme optimal, c'est le meilleur moyen pour vous perdre et rendre inutilement complexe votre code (vous pourrez une fois que vous avez une première version qui fonctionne) ;
- Tester au fur et à mesure (ex. vous pouvez créer votre propre fichier texte avec quelques cases au début pour vérifier que tout ou partie de votre code fonctionne bien) ;
- Réfléchir aux conditions (voir règle du jeu) avant de programmer le moteur de jeu ;
- Rester simple !

C. Rendu attendu

Le rendu est le fichier serpents.c ainsi complété et commenté. Ne mettre que des commentaires essentiels (ce qui est évident à la lecture du code ne nécessite pas de commentaire. Par exemple, éviter "`int i ; // déclaration de la variable entière i`").

Le code rendu doit compiler sans erreur et s'exécuter. Au besoin, fournir une version minimale fonctionnelle et incomplète plutôt qu'une version complète qui ne compile pas.

Le code est à **déposer** sur Chamilo dans l'outil travaux de la page du module S1103 pour, au plus tard, le **vendredi 18 février**.