.1 Piles et Files

Pile et Liste Chaînée par curseur : On utilise l'implémentation des listes chaînées vues dans le dernier TD en améliorant la gestion des places vides : elles seront gérées par une pile. On suppose que l'on possède les méthodes suivantes :

- `initPile()`
- `depile()`
- `empile(int newLibre)`
- `pileVide()`
- `pilePlein()`

1. En intégrant ces nouvelles données, redéfinir la méthode `inserListe(int position, char valeur)` vue en TD
2. Redéfinir également la méthode `supprimeListe(int position)`

File : Mise en oeuvre circulaire

1. Définir un type `FileTableauCirculaire` qui propose pour une file d'entier une implémentation contigüe par tableau circulaire
2. Quel est l'intérêt d'une telle implémentation ?
3. Écrire une méthode qui initialise la file avec une certaine longueur `int lng`
4. Écrire une méthode booléenne qui indique si la file est vide
5. Écrire une méthode booléenne qui indique si la file est pleine
6. Écrire une méthode qui ajoute un élément en queue de file
7. Écrire une méthode qui supprime l'élément en tête de file
8. Écrire une méthode qui donne la valeur de l'élément en tête de file

Double Pile : Trouver comment ranger deux piles dans un seul tableau `T` de sorte qu'on ne puisse plus empiler sur aucune que si le tableau est plein. Écrire une procédure `empiler(x,T,B)` qui empile l'élément `x` sur l'une des deux piles dans `T` selon la valeur du booléen `B`.

Tours de Hanoï : il s'agit d'un jeu mathématique basé sur le principe suivant.

- on dispose de N disques de diamètres différents
- on dispose de 3 "bases" sur lesquelles on peut empiler les disques
- on ne peut empiler un disque A sur un disque B que si le diamètre de A est inférieur à celui de B
- on ne peut déplacer qu'un seul disque à la fois
- au début du jeu tous les disques sont empilés sur la première base

Le but du jeu est de déplacer la totalité de la pile de disques de la première à la dernière base.

1. Représenter les états successifs des bases pour $N = 2$ puis $N = 3$.
2. Combien de déplacements faut-il pour terminer un jeu à N disques (raisonnez par récurrence) ?
3. Décrivez les structures de données adaptées pour représenter ce jeu.
4. Écrire le détail d'une fonction `DeplacerUnDisque(src,cible)` qui simule le déplacement d'un disque de la base `src` vers la base `cible`.
5. Donner un algorithme général de résolution du jeu qui soit *récursif* et utilise la fonction définie à la question précédente (indice : 5 lignes suffisent)