1 Exercice 1 : Création d'une liste composée des positions libres sur le damier

Le but de ce programme est le placement "protégé" de N dames sur un échiquier de NxN cases sans qu'aucune ne puisse être prise par une autre. Une dame peut prendre toutes les pièces qui se trouvent sur ses diagonales, ses lignes horizontale et verticale. Suivant les premiers emplacements choisis il n'est pas toujours possible de placer N dames. Nous allons travailler à partir d'une liste chaînée dont chaque élément contiendra :

- les coordonnées de la case (ou position) possible;
- un pointeur sur la case suivante.

Le programme devra donc comporter:

1. la structure de ces éléments : `Case`;
2. une fonction itérative `initListeCaseVideIter` qui permet de créer une liste composée de toutes les cases possibles du damier (ou échiquier) de manière itérative. Cette fonction renverra la tête de la liste ainsi créée;
3. une fonction récursive `initListeCaseVideRec` qui permet de faire la même chose (= créer une liste composée de toutes les cases possibles du damier (ou échiquier));
4. une fonction récursive `lengthListeRec` qui prend en paramètres la tête de la liste de cases et qui renvoie le nombre d'éléments (= cases) dans cette liste;
5. une fonction `rechercheAlea` qui prend en paramètres la tête de la liste de cases libres et qui renvoie les coordonnées d'une case libre choisies aléatoirement parmi les éléments de la liste;
6. une fonction `estPrise(i,j, xDame, yDame)` qui teste si un emplacement (i,j) est "prenable" par la dame de coordonnées (xDame, yDame);
7. une fonction `MAJ` qui prend en paramètres un pointeur sur la tête de la liste de cases libres ainsi que les coordonnées de l'emplacement de la nouvelle dame. Cette fonction va mettre à jour la liste des emplacements possibles, soit supprimer de la liste toutes les cases qui sont "prenables" par cette nouvelle dame;
8. une fonction `ajouteDame` qui prend en paramètres la tête de la liste qui contient les emplacements des dames et qui rajoute la position d'une nouvelle dame;
9. une fonction `main` qui permettra à partir d'un échiquier NxN de:
   - positionner le maximum de dames, donner ce maximum et donner leurs positions;
   - de recommencer jusqu'à ce que le nombre de dames positionnées soit égal à N ou que le nombre d'essais ait atteint sa valeur maximale. Vous afficherez le nombre d'essais exécutés avant de positionner N dames.

2 Exercice 2 : crible d'Eratosthène

Le programme demandé ici est de donner tous les nombres premiers inférieurs à N suivant le procédé du crible d'Eratosthène. Ce principe consiste à prendre la liste croissante des N premiers entiers en commençant par 1, et de supprimer successivement de la liste tous les multiples du 2e nombre encore présents dans la liste, puis les multiples du 3e, puis les multiples du 4e, ... jusqu'à arriver à la fin de la liste.

Quand le dernier entier restant dans la liste est atteint, alors la liste ne contient plus que les nombres premiers inférieurs à N.

Pour écrire ce programme vous allez utiliser une liste chaînée d'entiers que vous initialiserez au départ avec les N premiers entiers (N sera demandé à l'utilisateur), liste que vous mettrez à jour suivant le principe du crible d'Eratosthène afin de n'avoir à la fin plus que les entiers premiers inférieurs à N.

Vous afficherez la liste des nombres premiers ainsi que leur nombre.

3 Exercice 3 : dictionnaire

Le but de cet exercice consiste à manipuler un dictionnaire à l'aide de listes chaînées. On pourra rechercher et insérer des mots dans un dictionnaire, on pourra compter le nombre de mots du dictionnaire commençant par une lettre passée en paramètre.

Ce dictionnaire sera représenté par une liste chaînée d'éléments contenant un caractère, un pointeur vers une liste chaînée de mots commençant tous par ce caractère et un pointeur vers le prochain élément. Les éléments des listes seront insérés de sorte à ce que les éléments des listes soient triés dans l'ordre alphabétique.

La figure 1 illustre un dictionnaire de 7 mots commençant soit par 'a', soit par 'c', soit par 'l', soit par 'v':

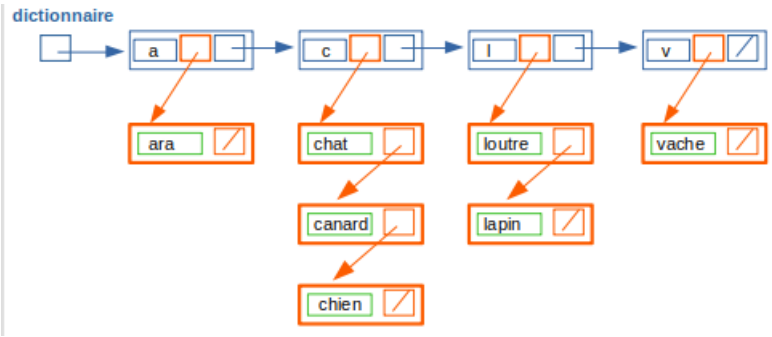

Figure 1: Exemple de dictionnaire

- Écrire une structure d'élément `mot` qui contient une chaîne de caratères (= string) et un pointeur vers un élément de type `mot`.
- Écrire une structure d'élément `index` qui contient un caractère (= char), un pointeur vers un élément de type `mot` et un pointeur vers un élément de type `index`.
- Écrire une fonction `rechercherIndex` qui permet de renvoyer l'adresse de l'index passé en paramètre dans la liste d'index également passée en paramètre. Si l'index n'existait alors il est inséré dans la liste (passée en paramètre) en respectant l'ordre alphabétique de la liste.
- Écrire une fonction `rechercherMot` qui permet de rechercher dans la liste d'index passée en paramètre, si le mot passé en paramètre existe. La fonction renverra vrai si le mot existe et faux sinon.
- Écrire une fonction `insererMot` qui permet d'insérér dans le dictionnaire (= liste d'index) passé(e) en paramètre, le mot (=string) passé en paramètre.
- Écrire une fonction `afficherMots` prenant 2 paramètres une liste chaînée d'éléments de type `index` et un caractère "car", qui affiche et compte tous les mots de la catégorie identifiée par "car" si elle existe.
- Écrire un programme principal, qui permet à l'utilisateur de réaliser plusieurs choses au choix:
  1. saisir et insérer dans le dictionnaire une série de mots terminée par -1;
  2. saisir et rechercher un mot;
  3. saisir un caractère et afficher le nombre de mots et les mots commençant par cette lettre;
  4. afficher le dictionnaire et le nombre de mots dans le dictionnaire.

  L'utilisateur pourra recommencer autant qu'il le souhaite.