Travaux Dirigés d'algorithmique n°6

Cours d'Informatique de Deuxième Année

—L2.1—

Listes chaînées par pointeurs

Définition de la structure :

Un élément d'une liste chaînée, appelé une *cellule*, contient les informations que l'on souhaite manipuler et également l'adresse de la cellule suivante. Les définitions de types suivantes mettent en œuvre la notion de "liste chaînée d'entiers" :

```
typedef struct cellule {
  int Valeur;                  /* donnee stockee : un entier.       */
  struct cellule * Suivant;   /* pointeur sur la cellule suivante. */
} Cellule;                     /* definition d'un nouveau type.     */

typedef Cellule * Liste;       /* definition d'un nouveau type.     */
```

L'intérêt de cette définition est qu'elle correspond à la définition mathématique, ainsi une liste est :

- soit vide ;
- soit un élément suivit d'un liste.

▸ **Exercice 1.** ***(Éléments d'une liste)***

- *Écrire une fonction qui prend en argument une liste chaînée et retourne le nombre d'éléments qu'elle contient.*
- *Écrire une fonction de recherche d'un élément dans une liste. La fonction renvoie l'adresse de la cellule qui contient l'élément s'il est présent et NULL sinon.*
- *Écrire une fonction qui prend en argument une liste chaînée et retourne le nombre d'éléments différents qu'elle contient.*
- *Donner la complexié des deux fonctions.*

▸ **Exercice 2.** ***(Affichage d'une liste)***

- *Écrire une fonction de nom* `AffListe` *qui affiche les éléments de la liste chaînée passée en paramètre. Vous donnerez une version itérative et une version récursive de cette fonction.*

– *Écrire une fonction de nom* `AffListeInv` *qui affiche, dans l'ordre inverse du chaînage, les éléments de la liste chaînée passée en paramètre.*
Pour la liste chaînée donnée auparavant, `AffListe` *devra produire la sortie :* `4 1 7 3` *et* `AffListeInv` *la sortie :* `3 7 1 4`.

▸ **Exercice 3. (Miroir)**
– *Écrire une fonction d'insertion d'une cellule en tête de liste*
– *Écrire une fonction d'extraction de la cellule en tête de liste.*
– *Écrire une fonction qui inverse l'ordre du chaînage des cellules d'une liste.*

▸ **Exercice 4.** ***(Libération d'une liste)***
Écrire une fonction `LibereListe` *qui libère tout l'espace mémoire occupé par une liste chaînée.*

▸ **Exercice 5. (Liste de mots)**
– *Définir les types nécessaires pour gérer des listes de mots.*
– *Parmi les fonctions précédentes, qu'elles sont celles qui peuvent être réutilisée sans transformations.*
– *Écrire la fonction d'insertion sans répétition pour ce type.*