TP 7 : Algorithmes sur les listes

Dans tout le sujet il est interdit d'utiliser les fonctions Maple qui font ce qu'on vous demande. On testera toutes les procédures écrites sur des exemples significatifs.

Exercice 1. Opérations élémentaires

1. Écrivez une procédure `Maxi` qui prend en argument une liste d'entiers `L` et renvoie le maximum des éléments de cette liste.
2. Écrivez une procédure `Cherche` qui prend en argument une liste `L` et un élément `a` et renvoie un indice `i` tel que `L[i]=a` s'il existe, et -1 sinon.
3. Écrivez une procédure `ChercheDichotomie` qui fait la même chose que la précédente mais en supposant que la liste en argument est triée dans l'ordre croissant. On remarquera qu'on peut regarder l'élément au milieu de la liste, et suivant s'il est plus petit ou plus grand que celui qu'on cherche on sait dans quelle moitié de la liste il suffit de chercher.
4. Écrivez une procédure `Retourne` qui prend en argument une liste `L` et renvoie la même liste dans l'ordre inverse.
5. Écrivez une procédure `Applique` qui prend en argument une liste `L` et une procédure `f`, et qui renvoie la liste des éléments de `L` auxquels on a appliqué `f`. Par exemple, `Applique([1,-1,7],x->x^2)` doit renvoyer `[1,1,49]`.
6. Écrivez une procédure `Supprime` qui prend en argument une liste `L` et un entier `i` et qui renvoie la liste dans laquelle on a supprimé le `i`-ème élément.
7. Écrivez une procédure `Insere` qui prend en argument une liste `L`, un entier `i` et un élément `a` et qui renvoie la liste dans laquelle on a inséré l'élément `a` juste après le `i`-ème élément.
8. Écrivez une procédure `Echange` qui prend en entrée une liste `L` et deux entiers `i,j`, et qui renvoie la liste dans laquelle on a échangé les valeurs aux indices `i` et `j`. Par exemple, la commande `Echange([25,31,44,12],1,3)` doit renvoyer `[44,31,25,12]`.

Exercice 2. Structures de données à base de liste

1. Une file est une structure de données dans laquelle on peut ajouter et retirer des éléments, et dans laquelle les éléments qui sont ajoutés en premier sortent aussi en premier (penser à une file d'attente). On va implémenter une file à l'aide de listes, les éléments arrivés en premier seront au début de la liste, les éléments arrivée en dernier seront à la fin de la liste.
   a) Écrivez une procédure `CreeFile` qui ne prend aucun argument et qui renvoie une liste représentant la file vide.
   b) Écrivez une procédure `EstVideFile` qui prend en argument une liste `L` représentant une file et qui renvoie `true` ou `false` selon que la file est vide ou non.
   c) Écrivez une procédure `AjouteFile` qui prend en argument une liste `L` représentant une file et un élément `a` et qui renvoie la liste représentant la file à laquelle on a ajouté `a`.
   d) Écrivez une procédure `SortieFile` qui prend en argument une liste `L` représentant une file et qui renvoie le prochain élément qui doit sortir de la file.

e) Écrivez une procédure `RetireFile` qui prend en argument une liste `L` représentant une file et qui renvoie la liste représentant la file à laquelle on a retiré l'élément qui doit sortir.

2. Une file est une structure de données dans laquelle on peut ajouter et retirer des éléments, et dans laquelle les éléments qui sont ajoutés en premier sortent en dernier (penser à une pile de livres). On va implémenter une pile à l'aide de listes, les éléments arrivés en premier seront au début de la liste, les éléments arrivée en dernier seront à la fin de la liste. Écrivez les procédures `CreePile`, `EstVidePile`, `AjoutePile`, `SortiePile`, `RetirePile` réalisant les mêmes actions que les précédentes mais pour une pile.

Exercice 3. Tris

Le but est d'implémenter différents algorithmes pour trier une liste dans l'ordre croissant.

1. On appelle *tri par sélection* l'algorithme qui consiste à chercher le plus petit élément de la liste, puis à l'échanger avec le premier, puis chercher le plus petit parmi ceux qui restent, et l'échanger avec le second, etc. Écrivez une procédure `TriSelection` qui prend en argument une liste `L` et qui renvoie la liste triée en ordre croissant en utilisant l'algorithme de tri par sélection.
2. L'algorithme de *tri par insertion* consiste à maintenir à chaque étape k les k premiers éléments de la liste triés, à chercher où doit aller le $(k+1)$-ème élément et à l'insérer à sa position. Écrivez une procédure `TriInsertion` qui prend en argument une liste `L` et qui renvoie la liste triée en ordre croissant en utilisant l'algorithme de tri par insertion.
3. L'algorithme de *tri rapide* consiste à choisir un élément le la liste qu'on appelle le *pivot*, à parcourir la liste de sorte à mettre tous les éléments plus petits que lui avant lui, tous les éléments plus grands après, puis à appeler le tri récursivement sur les listes de gauche et de droite. Écrivez une procédure `TriRapide` qui prend en argument une liste `L` et qui renvoie la liste triée en ordre croissant en utilisant l'algorithme de tri rapide.
4. L'algorithme de *tri fusion* consiste à couper la liste en deux par le milieu, puis à trier chacune des deux listes obtenues en appelant le tri récursivement, puis à recréer la liste totale triée à partir des deux sous-listes (*fusion* des sous-listes). Pour cela il suffit de remarquer que le premier élément de la liste fusionnée est le premier élément de l'une des listes, puis qu'on peut répéter ce procédé en supprimant cet élément qu'on a sélectionné.
   a) Écrivez une procédure `Fusion` qui prend en argument deux listes `L1` et `L2` supposées triées en ordre croissant, et qui renvoie la liste triée contenant les éléments des deux listes.
   b) Écrivez une procédure `TriFusion` qui prend en argument une liste `L` et qui renvoie la liste triée en ordre croissant en utilisant l'algorithme de tri fusion.