Pratique de la programmation et projet
TP 10 : Listes doublement chaînées

Frédéric Vivien

Listes doublement chaînées

Nous nous intéressons maintenant aux listes doublement chaînées, c'est-à-dire chaînées dans les deux sens : en plus du champ *successeur*, chaque élément contient un champ *prédécesseur* qui est un pointeur sur l'élément précédent dans la liste. Définissez un tel type de liste et écrivez pour ce type des fonctions équivalentes à celles demandées pour les listes simplement chaînées, toutes les fonctions renvoyant des listes devant renvoyer des listes doublement chaînées :

1. `is_empty` : teste si une liste est vide.
2. `car` : retourne la valeur contenue dans la tête de la liste argument.
3. `cdr` : retourne le reste de la liste argument.
4. `longueur` : calcule la longueur d'une liste.
5. `print_list` : affiche le contenu d'une liste.
6. `print_list_double` : affiche le contenu d'une liste de la tête à la queue puis de la queue à la tête.
7. `appartient` : teste l'appartenance d'un élément à une liste.
8. `egal` : test l'égalité de deux listes.
9. `dernier` : renvoie la valeur du dernier élément de la liste.
10. `cons` : rajoute un élément en tête d'une liste.
11. `rplaca` : remplace la valeur en tête de la liste.
12. `rplacd` : remplace le reste de la liste.
13. `copy` : crée une liste copie de la liste argument.
14. `concat` : concatène deux listes, vous écrirez deux versions de cette fonction, l'une avec remplacement physique et l'autre sans (autrement dit, l'une des deux versions modifie les listes passées en argument, mais pas l'autre version).
15. `reverse` : renverse une liste (écrivez deux versions de cette fonction : une avec et une sans accumulateur).