TP 6 SDA Listes simplement chaînées

Implémenter les listes d'entiers simplement chaînées:

Structure de donnée:

Noeud: noeud de la liste (valeur + chaînage)

Liste: pointeur vers le 1er noeud + nb de noeuds + ??

Fonction à implémenter (et à tester):

- Liste* liste_vide() : génère une liste vide
- Noeud* push_front(Liste* l, int v) : ajout en tête
- void pop_front(Liste* l) : retirer la tête
- int front_val(Liste* l) : valeur en tête
- void print(Liste* l) : affiche la liste
- Bool est_vide(Liste* l): teste si la liste est vide
- Noeud* trouve_premier(Liste* l, int v)
- int occurence(Liste* l, int v) : compte le nombre de *v* dans *l*.
- Noeud* insert_after(Noeud* c, int v) : insert une valeur après un noeud
- void retire(Liste* l, Noeud* n) : retire un noeud de la liste
- Noeud* push_back(Liste* l, int v) : ajout en fin de liste
- void pop_back(Liste* l) : retire le dernier noeud de la liste
- int back_val(Liste* l) : dernière valeur de la liste

Penser aux cas particuliers (liste vide, un seul élément).

On pourra implémenter des fonctions utiles comme:

- Noeud* nouveau_noeud(int v)

Il est fortement recommandé de faire un(des) dessin(s) pour chaque fonction avant de coder.

Sachant qu'un pointeur de fonction à la syntaxe suivante:

type_retour (*fonction)(paramètres)

et qu'il suffit d'écrire son nom pour récupérer le pointeur.

Ecrire une fonction apply qui applique une fonction à tous les éléments d'une liste.

Tester avec apply(l,foisdeux); (je vous laisse deviner ce que fait la fonction foisdeux !