TP 6 bis SDA Listes doublement chaînées ...

1 Implémenter les listes d'entiers doublement chaînées:

Structure de donnée:

Noeud: noeud de la liste (valeur + chaînages)

Liste: pointeur vers le 1er noeud + dernier 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
- 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
- void print(Liste* l) : affiche la liste
- Noeud* trouve_premier(Liste* l, int v)
- Noeud* trouve_dernier(Liste* l, int v)
- void retire(Liste* l, Noeud* n) : retire un noeud de la liste
- Noeud* insert_after(Noeud* c, int v) : insert une valeur après un noeud

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

2 Faire un (deux) programmes de benchmark de listes simplement / doublement chaînées

On testera en faisant 100000 push_front, 100000 pop_back, 100000 push_back, 100000 pop_front. (trouver le bon chiffre pour que le prog. s'exécute en quelques secondes)

Vous utiliserez la compilation séparée:

- fichiers liste_simple.h et liste_simple.c
- fichiers liste_double.h et liste_double.c
- fichier (programme) benchmark_simple.c
- fichier (programme) benchmark_double.c

3 Implémenter les listes d'entiers dans un tableau

- commencer par une version ou on alloue une taille max au départ
- faire une version ou l'on réalloue (paquet de 10000) quand c'est nécessaire.
- faire les *benchmark* correspondant
- changer le test pour ne faire que des push et pop *front*.