TD/TP Les listes linéaires

Exercice n°1

1. Programmer les opérations suivantes définies sur les listes linéaires unidirectionnelles : (représentation contiguë) représentation chainée

   - ✓ Créer une liste linéaire
   - ✓ Initialiser une liste linéaire
   - ✓ Tester si la liste est vide
   - ✓ Supprimer le successeur d'un élément référencé
   - ✓ Supprimer un élément référencé
   - ✓ Supprimer le premier élément
   - ✓ Supprimer le dernier élément
   - ✓ Insérer (ou ajouter) un élément avant le premier élément
   - ✓ Insérer un élément après le dernier élément
   - ✓ Parcourir (afficher) les éléments de la liste

2. Traduire en C les opérations précédentes.

   Écrire un algorithme qui gère deux listes contenant des entiers positifs.

Exercice n°2

1. Programmer les opérations définies sur une liste linéaire bidirectionnelle.
2. Traduire en C les opérations précédentes.

Exercice n°3

Dans ce problème, on suppose que N personnes vont jouer un jeu de groupe ; pour cela elles forment un cercle, la M^{ième} personne sur ce cercle sort de cette dernière, et le cercle se reforme sur les survivants à chaque fois. Le problème est de déterminer qui sera la dernière à rester, ou plus généralement de trouver l'ordre dans lequel les personnes quittent le cercle.
Ecrire un algorithme permettant de lire N et M et d'afficher l'ordre dans lequel les personnes sortent du cercle.
Exemple : Pour N=9 et M=5, la liste s'opère dans l'ordre de 5, 1, 7, 4, 3, 6, 9, 2, 8.

Exercice n°4

Ecrire une procédure de tri d'une liste chaînée unidirectionnelle.

Exercice n°5

Écrire un algorithme de gestion des étudiants comme suit :
1 – Ajout d'un étudiant.
2 – Rechercher d'un étudiant.
3 – Suppression d'un étudiant.
4 – Tri de liste des étudiants.
5 – Affichage de liste des étudiants.
6 – calculer et afficher l'âge moyen des étudiants considérées.

Un étudiant est représenté par la structure suivante :
Type ETUDIANT = **enregistrement**
{ nom : chaine[50] ; // nom et prénom d'un étudiant
Adr : chaine [80] ; // adresse d'un étudiant
ID : entier ; //numéro d'inscription
}

Exercice 7

Dans cet exercice, on s'intéresse à traiter une liste simplement chaînée de réels.
1) Ecrire une fonction Longueur qui permet de calculer la longueur de la liste
2) Ecrire une procédure **MinMax** qui permet de calculer le min et le max dans une liste chaînée de réels.
3) Ecrire une procédure **MoyProd** qui permet de calculer la moyenne et le produit d'une liste chaînée de réels.
4) Écrire une fonction qui permet de fusionner deux listes chaînées d'entiers en une troisième liste telle que celle-ci contienne les éléments de la *liste1* et les éléments de la *liste2*.

Exercice 8

Un abonné est caractérisé par :
- son numéro
- son nom
- sa ville

1) Ecrire une fonction qui permet de vérifier s'il existe dans la liste un abonné dont le nom est N et la ville est V.
2) Ecrire une fonction qui permet de calculer le nombre d'abonnés d'une ville donnée.