Structures de données en C

TD/TP 1 – Les liste chainées

Exercice 1 : Rechercher l'élément maximal d'une liste simplement chaînée

Écrire deux fonctions, l'une itérative, l'autre récursive, qui permettent de déterminer la valeur maximale d'une liste chaînée d'entiers positifs.

Exercice 2 : Concaténer deux listes

On considère deux listes chaînées L1 et L2 dont les éléments sont des entiers. Écrire une fonction qui rattache la liste L2 à la suite de la liste L1. Tous les cas particuliers doivent être pris en compte.

Exercice 3 : Extraire deux listes à partir d'une liste

Soit une liste de nombres **relatifs**. Écrire une fonction permettant de séparer cette liste en deux listes : la première ne comportant que des **entiers** positifs ou nuls, et la seconde ne comportant que des nombres **entiers** négatifs.

Exercice 4 : Permuter deux places d'une liste

Écrire une fonction sur une liste simplement chaînée qui échange les positions des cellules données par deux pointeurs t et v.

Exercice 5 : Supprimer des éléments

Soit L une liste chaînée. Écrire trois fonctions tels que :

1. Dans la première on supprime toutes les occurrences d'un élément donné x.
2. Dans la deuxième, on ne laisse que les k premières occurrences de cet élément et on supprime les suivantes.
3. Dans la troisième, pour chaque élément de la liste, on ne laisse que sa première occurrence.

Concevez la seconde fonction en adaptant la première, puis concevez la dernière en exploitant la seconde.

Exercice 6 : Inverser une liste

Écrire un algorithme qui inverse une liste chaînée sans recopier ses éléments. Réaliser une version itérative et une autre récursive.

Exercice 7 : Refermer une liste sur elle-même

Concevoir un algorithme qui transforme une liste simplement chaînée linéaire en liste simplement chaînée circulaire.

Exercice 8 : Saisir, enregistrer puis évaluer un polynôme

Enregistrer les coefficients et les exposants d'un polynôme dans une liste chaînée. Évaluer ce polynôme pour une valeur donnée de x.
Utiliser au moins deux fonctions dans ce programme :

1. L'une pour construire la liste sur la base des coefficients et des exposants, qui sont saisis au clavier.
2. La seconde afin d'évaluer le polynôme pour la valeur x, également saisie au clavier.

Exercice 9 : Construire une liste circulaire ordonnée

Écrire une fonction qui créée une liste circulaire ordonnée d'entiers à partir d'un tableau non ordonné d'entiers.

Exercice 10 : Réaliser le chaînage arrière d'une liste doublement chaînée

Concevoir une fonction qui réalise le chaînage arrière d'une liste doublement chaînée dont seul le chaînage avant a été effectué.

Exercice 11 : Inverser pile et file

Concevoir deux fonctions, qui créent respectivement :

1. La file inverse d'une file.
2. La pile inverse d'une pile.

Ces deux fonctions doivent restituer leur entrée inchangée.

Exercice 12 : Simuler la récursivité à l'aide d'une pile

Écrire une fonction pour calculer la somme de 1 à n avec $n \in N^*$ en simulant la récursivité à l'aide d'une pile.
Les en-têtes des opérations à utiliser pour cet exercice sont fournis :

- FONCTION nouvellePile() : pile
- FONCTION estPileVide(p : pile) : booléen
- FONCTION empiler (val : T, *pp : pile)
- FONCTION depiler (*pval : T, *pp : pile) : entier

Exercice 13 : Problème de Joseph

Écrire une fonction permettant de lire deux entiers positifs n et k. Construire une liste circulaire dans laquelle seront enregistrés les nombres 1, 2, 3, ..., n, dans cet ordre. En commençant à partir du nœud contenant 1, supprimer successivement tous les $k^{ièmes}$ nœuds de la liste, en effectuant un parcours circulaire dans celle-ci et jusqu'à ce que tous les nœuds aient été supprimés. Dès qu'un nœud est supprimé, le suivant dans la boucle est considéré comme la nouvelle tête de liste et ainsi de suite. Si n = 8 et k = 3, par exemple, la suppression des nœuds contenant les huit entiers se fait dans cet ordre :
3, 6, 1, 5, 2, 8, 4, 7
(Ce procédé peut être illustré par un groupe composé à l'origine de n personnes formant un cercle, que l'on élimine successivement en désignant le $k^{ièmes}$ de ceux restant dans le cercle que l'on continue de parcourir).