École Polytechnique Sousse
Département Informatique
Section : Génie Informatique
Niveau : 3^{eme} Année–AU : 2022-2023

Algorithmique et Structures de Données
Travaux Dirigés N°2

Exercice 1

Nous partons de la définition d'une liste linéaire bidirectionnelle ci-après :

```
cellule = structure
        avant     : ^cellule
        info      :  entier
        suivant   : ^cellule
Fin structure
liste = structure
        premier   : ^cellule
        dernier   : ^cellule
Fin structure
VAR   :
      L : liste
```

1. Écrire une procédure INVERSER_LISTE permettant d'inverser une liste L.
2. Écrire une procédure INSERTION_VAL_AVANT qui prend en entrée une liste L et une valeur x et qui insère avant chaque élément de la liste la valeur x.
   Par exemple, x = 1, liste initiale L = 2, 7, 10 après l'insertion de 1 après chaque élément, la liste devient L= 1, 2, 1, 7, 1, 10.
3. Écrire une procédure SUPPRIMER_DERNIERE_OCC qui supprime dans une liste la dernière occurrence d'un élément x donné.
4. On se propose de construire une liste L3 contenant tous les éléments communs des deux listes linéaires bidirectionnelles L1 et L2 en éliminant les redondances.
   Écrire les procédures/fonctions nécessaires pour résoudre le problème cité ci dessus.
   Exemple
   L1 = $\{10, 50, 100, 23, 11, 50, 23\}$
   L2 = $\{200, 23, 17, 99, 10, 50\}$
   Résultat :
   L3 = $\{10, 50, 23\}$

Exercice 2

On se propose de traiter les pistes d'un CD-ROM audio, chaque piste correspond à une chanson. Pour ce faire, on stockera le contenu du CD dans une liste linéaire bidirectionnelle. Chaque piste est caractérisée par son code, le titre, l'artiste, la durée de la chanson.

1. Définir les différentes structures en utilisant une liste linéaire bidirectionnelle.
2. La création de la liste permet de saisir une liste de N pistes avec les données nécessaires relatives à chaque chanson. Pour cala écrire une procédure REMPLIR qui permet de créer une liste bidirectionnelle de pistes L1.
3. Écrire une procédure AFFICHE qui permet d'afficher la liste L1 des pistes.
4. Une fonction MIN_PISTE permettant de chercher et retourner l'adresse de la piste ayant une durée minimale de la liste L1.
5. Une procédure SUPP_CHANSON permettant de supprimer une piste pointée par un pointeur pmin de la liste L1. N'oublier pas de traiter les cas particuliers.
6. Une procédure TRI permettant de trier la liste L1 dans une nouvelle liste L2 en respectant le principe suivant :
   — chercher l'adresse de la piste ayant la durée minimale de la liste L1,
   — Insérer la piste cherchée en tête dans la liste L2,
   — Supprimer la piste cherchée de la liste L1,
   — Répéter ce traitement jusqu'à ce que la liste L1 soit vide et L2 une liste triée par ordre décroissant selon la durée.
   — Retourner la liste L2
7. Écrire le programme principal :
   — Charger une liste linéaire bidirectionnelle de pistes L1,
   — Afficher la liste L1,
   — Créer une liste linéaire bidirectionnelle de pistes L2 triée à partir de L1.
   — Afficher la liste L2.