ATELIER 5: LES LISTES CHAINEES

Exercice 1:

1. Ecrire une fonction ***Saisir*** permettant de remplir un tableau **T** avec **N** entiers.
2. Ecrire une fonction ***Copier*** qui permet de copier un tableau de **N** entiers dans une liste chaînée.
3. Ecrire une fonction ***AfficherPaire*** permettant d'afficher les nombres pairs d'une liste chainée d'entiers.
4. Ecrire un programme principal permettant de tester ces fonctions.

Exercice N°2:

On se propose de modéliser la gestion des patients dans un cabinet médical. Un patient est caractérisé par: le ***nom***, le ***prénom*** et un champ ***rdv*** (pour rendez-vous) de type entier indiquant si le patient a un rendez-vous ou pas: 0 si le patient est sans rendez-vous, 1 si la patient est avec rendez-vous.
Avant d'être consultés par le médecin, les patients sont entrés dans une salle d'attente qui sera modélisée par une liste chaînée de patients. Une secrétaire fait entrer les patients ayant un ***rdv=1*** selon leur ordre d'arrivée, ensuite elle fait entrer les autres patients (ceux dont le ***rdv=0***) selon leur ordre d'arrivée aussi.

Définir la structure de données ***Patient***.
Définir la structure de données ***Cellule***.

Dans une deuxième étape, on vous demande d'écrire les fonctions suivantes :

1. ***Cellule * AjoutPatient (Cellule *tete, Patient P)*** , qui permet d'ajouter un nouveau patient à la fin de la liste identifiée par son pointeur tête.
2. ***void RendezVous (Cellule *tete, int *rdv, int *sansRdv)***, qui compte et retourne le nombre de patients avec RDV, et le nombre de patients sans rendez-vous.
3. ***Cellule * SupprimePatient (Cellule *tete)*** , qui permet de faire entrer un patient en consultation. Cette opération est effectuée de la manière suivante : s'il n'y a aucun patient avec rendez-vous alors c'est le premier patient de la liste qui est supprimé. Sinon on supprime le premier patient qui a un rendez-vous.
4. ***void ConsulterSalleAttente (Cellule *tete)*** , qui affiche tout d'abord les patients avec rendez-vous, ensuite les patients sans rendez-vous.

Exercice N°3:

On veut gérer à travers une liste chaînée un ensemble de comptes. Pour chaque compte on garde :

- ✓ le numéro du compte (entier)
- ✓ le nom du propriétaire (30 caractères)
- ✓ le solde (de type réel)

Faite la déclaration de la structure ***Compte*** et ***Cellule*** nécessaire pour notre liste puis répondre à ces questions:

1. Ecrire une fonction ***AjouterCompte*** qui ajoute un compte à la fin de la liste (uniquement si le compte possède un solde supérieur ou égal à zéro).
2. Ecrire une fonction ***SoldeCompte*** qui cherche un compte en fonction de son numéro et retourne son solde s'il existe dans la liste ou bien -1 s'il n'existe pas.
3. Ecrire une fonction ***Total*** qui calcule et retourne la somme des soldes de tous les comptes de la liste.
4. Ecrire une fonction ***CompteMax*** qui retourne le compte qui possède le solde maximal de tous les comptes de la liste.
5. Ecrire une fonction ***Rechercher*** qui consiste à chercher un compte en fonction du nom de propriétaire dans la liste.
6. Ecrire une fonction ***Afficher*** qui affiche tous les comptes dans la liste.

Exercice N°4:

Une commande est composée de plusieurs lignes. Une ligne de commande est définie par :

- Un numéro de commande (**numCmd**): un entier.
- Une référence d'article (**refArt**): une chaîne de caractères.
- Une quantité commandée (**qtArt**): c'est le nombre d'articles commandés.
- Un prix unitaire d'un article (**prixArt**): un réel.

- Définir la structure **LigCom**. (Ligne de Commande) Définir la structure de données **Cellule** qui décrit un noeud de la liste.

On dispose d'une liste pour gérer les commandes, la liste est composée par les lignes des commandes regroupées par commandes c.à.d les lignes qui ont le même numéro de commande doivent se présenter en séquence (les unes après les autres).

Il s'agit de développer en C les fonctions suivantes:

- ***Cellule* ajouter (Cellule *tete, LigCom L):*** Ajoute une ligne de commande à la liste des lignes de commandes. Si la liste contient déjà des lignes de la même commande, la ligne doit être ajoutée au début de la séquence des lignes sinon à la fin de la liste.
- ***float totalCommande (Cellule *tete, int numCmd):*** qui prend en paramètre la liste des commandes et le numéro d'une commande et retourne le total de la commande (somme de qtArt * prixArt des lignes de la commande).
- ***Cellule* supprimerLigne (Cellule *tete, char refArt[]):*** qui prend en paramètre une référence d'un article et la liste des commandes et qui supprime la première cellule avec la référence donnée.
- ***void afficherCommande (Cellule *tete, int numCmd):*** qui prend en paramètre la liste des commandes et le numéro d'une commande et affiche ses lignes ainsi que le total de la commande.