Série N° 2

Exercice 1 :

On désire définir un type abstrait permettant de manipuler un nombre complexe. Ce type abstrait sera appelé Complex. Le type défini devra comporter des opérations suivantes :

- **constructeur** d'un nombre complexe z=a + bi à partir de deux réels a et b ;
- **re** et **im** qui détermine respectivement la partie réelle et la partie imaginaire d'un nombre complexe ;
- **add** et **mult** pour additionner et multiplier deux nombres complexes ;
- **module** et **conj** pour le module et le conjugué d'un nombre complexe,

1- Spécification abstraite : Spécifiez complètement le type abstrait Complex.
2- Spécification opérationnelle : Proposez une structure de données permettant de représenter le type abstrait Complex.

Exercice 2 :

On veut réaliser une application de gestion d'une bibliothèque. Pour cela chaque livre est caractérisé par les informations suivantes :

- le numéro du livre (**num)** : un entier positif,
- le titre du livre (**titre**) : une chaîne de 20 caractères,
- le nom de l'auteur (**auteur**) : une chaîne de 10 caractères,
- le nom de l'éditeur (**editeur**) : une chaîne de 10 caractères,
- l'année de son édition (**annee**) : un entier,
- le prix du livre (**prix**) : un réel,
- le nombre d'exemplaires dans le stock (**ne**) : un entier.

On souhaite gérer une bibliothèque qui contient un nombre de livres non prédéfini. On veut de plus concevoir une structure de données dynamique permettant d'organiser ses informations dans la mémoire. Les opérations prévues sur cette structure sont :

- l'opération qui permet de créer une liste vide : **listeVide**() ;
- l'opération qui ajoute un livre au début de la liste : **ajouterLivre**(liste, livre) ;
- l'opération qui vérifie si un livre est dans la bibliothèque ou non : **existeLivre**(liste, livre), la fonction retourne le numéro du livre s'il existe dans la liste et -1 sinon ;
- l'opération qui renvoie le nombre de livres d'un éditeur donné : **nbLivresEditeur**(liste, editeur) ;
- l'opération qui renvoie le livre le plus cher : **livrePlusCher**(liste)
- l'opération qui renvoie la liste des livres d'un auteur donné: **livresAuteur**(liste, auteur) ;
- l'opération qui retourne la liste des livres dont le nombre d'exemplaires égal à 0 (Livres épuisés dans le stock) : **livresEpuises**(liste) ;
- l'opération qui renvoie la liste de livres triés par ordre décroissant de l'année d'édition : **trierLivresAnnee**(liste) ;

- l'opération qui décrémente le nombre d'exemplaire d'un livre dans la liste par nb le nombre vendu : **vendre**(liste, num, nb) ;

1) Donnez la spécification abstraite de type « Bibliothèque ».

2) On veut implémenter la spécification opérationnelle de ce type abstrait. Pour cela :

a. Expliquez pourquoi une liste chaînée est une solution élégante à ce problème.
b. Ecrivez le type livre qui représente correctement un livre.
c. Ecrivez la structure Maillon d'une telle liste, chaque maillon représentant un livre de la bibliothèque et un pointeur sur le suivant.
d. Donnez une implémentation des opérations précédentes.

Exercice 3 :

Une entreprise souhaite une application de gestion de stock de ses produits.
Un produit est caractérisé par :

- **nom** : libellé du produit
- **qnt** : la quantité disponible
- **prixU** : le prix unitaire du produit.

Dans ce problème, on propose de définir une structure de données « stock »permettant la gestion de stock. Les opérations prévues sur la structure de données stock sont :

- **creerStock**() : permet de créer une structure vide.
- **existe**(st, lib) : permet de vérifier l'existence d'un produit dont le libellé est passé en paramètre dans la structure stock 'st'. Elle retourne l'adresse du maillon contenant le produit 'lib' s'il existe sinon elle retourne NULL.
- **acheter**(st, nb, lib, nb, prix) : permet d'alimenter la structure stock 'st' par 'nb' unités du produit 'lib' ayant le prix unitaire 'prix'. Si le produit existe déjà dans la structure, l'opération d'ajout "acheter" augmente sa quantité par nb.
- **supprimer**(st, lib) : permet de supprimer un produit, s'il existe, dont le libellé est passé en paramètre dans la structure stock 'st'.
- **vendre**(st, lib, nb) : permet de réduire la quantité d'un produit de nom 'lib' dans le stock 'st' par le nombre d'unités vendues 'nb'. Attention : Il faut supprimer du stock le produit dont la quantité atteint 0.
- **prixStock**(st) : permet de calculer le prix total des produits dans le stock st.

1. Donnez la spécification abstraite de la structure de données stock.
2. Spécification opérationnelle : On veut représenter la structure stock sous forme d'une liste doublement chaînée.
   a. Donnez entité maillon nécessaires à la représentation de cette liste ;
   b. Ecrivez les opérations creerStock, acheter, existe, supprimer, vendre et prixStock.