Exercice 1 : Nombres complexes et Diagrammes de Bode

Dans cet exercice on s'intéresse à la représentation de la réponse en fréquence d'un filtre (par exemple ceux des systèmes électriques). La représentation du comportement d'un filtre en régime asymptotique peut se faire au moyen d'une fonction $H(\omega)$ d'une variable réelle (la fréquence) à valeurs dans les complexes. La norme $|H(\omega)|$ représente le gain du filtre (le ratio, amplitude de sortie par rapport à l'amplitude d'entrée), l'angle $\arg(H(\omega))$ représente la phase du filtre (le déphasage entre le signal de sortie et le signal d'entrée). Dans tout l'exercice l'unité complexe sera notée j.

1. Représentation des nombres complexes

Q. 1 Définir une structure `struct complex;` permettant de représenter un nombre complexe z au moyen de deux `float`, `x` et `y` tel que $z = \texttt{x} + j\texttt{y}$.

Q. 2 Définir une fonction `void print_complex(struct complex c);`, affichant un nombre complexe

Q. 3 Définir des fonctions `add`, `mul`, `div` calculant l'addition, la multiplication et la division de deux nombres complexes. À titre d'exemple la fonction `add` devra avoir la signature `struct complex add(struct complex c1, struct complex c2);`.

Q. 4 Définir une fonction `scal` permettant le calcul d'un produit $\alpha.\texttt{z}$ où `z` est un complexe et α est un nombre flottant.

Q. 5 Définir un nouveau type structuré `struct complex2` contenant deux flottants `r` et `theta` et représentant le nombre complexe $\texttt{r}e^{j\texttt{theta}}$.

Q. 6 Définir une fonction `void print_complex2(struct complex2 c);`, affichant un nombre complexe

Q. 7 Définir deux fonctions de conversion `struct complex to_complex(struct complex2 c);` et `struct complex2 of_complex(struct complex c);` permettant les traductions d'une représentation en l'autre.

2. Filtres

On s'intéresse à la manipulation de filtres d'ordre 2, qui peuvent être mis sous la forme :

$$\begin{aligned} H(\omega) &= \frac{P_1(\omega)}{P_2(\omega)} \\ P_1(\omega) &= a_1 + b_1(j\omega) + c_1(j\omega)^2 \\ P_2(\omega) &= a_1 + b_2(j\omega) + c_2(j\omega)^2 \end{aligned}$$

où les a_i sont des nombres réels.

Q. 8 Définir un type structuré permettant la représentation d'un polynôme d'ordre 2 à coefficients flottants.

Q. 9 En déduire un type structuré permettant la représentation d'un filtre d'ordre 2 en tant que 2 polynômes d'ordre 2.

Q. 10 Définir le filtre passe-bas d'ordre 2 suivant : $H(\omega) = \frac{1}{1+(j\omega)+(j\omega)^2}$. Définir le filtre passe-bande d'ordre 2 suivant : $H(\omega) = \frac{1+(j\omega)^2}{1+(j\omega)+(j\omega)^2}$.

Q. 11 Définir une fonction d'évaluation `eval` prenant un argument un filtre H et une fréquence ω et calculant $H(w)$.

Q. 12 En déduire deux fonctions `dephasage` et `gain` prenant en arguments un filtre H et une fréquence ω et retournant le déphasage et le gain du filtre H sur la fréquence ω.

Q. 13 Définir une fonction `plot_bode` prenant en argument deux entiers l et u ainsi qu'un filtre H et imprimant à l'écran une ligne par entier i de l'intervalle $[\![l, u]\!]$, le déphasage du filtre H sur la fréquence 10^i et le gain du filtre H sur la fréquence 10^i, séparées par des tabulations (`"\t"`). On peut choisir le nombre de digits à faire afficher dans un flottant au moyen du motif (par exemples pour 10 digits) : `"%.10f"`. On testera la fonction `plot_bode` au moyen des deux filtres exemples définis ci-avant.

Exercice 2 : Base de données d'étudiants

Dans cet exercice on s'intéresse à une base de données d'étudiants inscrits à l'université.
Un étudiant sera représenté par 4 grandeurs : son nom (une chaîne de caractères), son prénom (une chaîne de caractères), son numéro étudiant, ses notes. Les notes d'un étudiant sont un tableau de 6 flottants, chaque flottant représentant la note dans une des 6 matières de l'étudiant.
Une matière est représentée par 3 grandeurs : son nom, son identifiant, son seuil de validation. Si l'identifiant d'une matière m est i alors la case d'indice i du tableau de notes d'un étudiant contient sa note dans la matière m. Le seuil de validation représente la note minimale qu'il faut avoir pour valider la matière. Dans tout l'exercice les chaînes de caractères seront stockées dans des tableaux de 50 caractères.

Q. 1 Définir un type structuré `etudiant` (resp. un type structuré `matiere`) permettant la représentation d'un étudiant (resp. d'une matière).

On donne le tableau d'étudiants suivant :

Prénom	Nom	Id	Note 0	Note 1	Note 2	Note 3	Note 4	Note 5
Ada	Lovelace	12	10.5	11.5	9.5	8.5	7.	14.
Alan	Turin	25	15.	12.	8.	7.	13.	9.
Margaret	Hamilton	38	18.	12.	5.	17.	16.	8.
Alonzo	Church	1	5.	12.	13.5	17.	16.5	10.5
Radhia	Cousot	3	17.	19.	18.	15.	14.	17.

Et le tableau des 6 matières suivantes :

Nom	Id	Seuil
Anglais	5	12
Mathématiques	1	12
Informatique	0	8
Physique	2	10
Français	4	10
SI	3	11

Q. 2 Définir deux variables globales : `table_etu_exemple` et `table_matiere_exemple` comme étant les deux tableaux ci-dessus (un tableau de 5 étudiants et un tableau de 6 matières). On utilisera ces deux tableaux pour tester les fonctions du reste de l'exercice.

Q. 3 Définir une fonction `bool eq_str(char s1[50], char s2[50])` testant l'égalité des deux chaînes de caractères passées en argument.

Q. 4 Donner une fonction `int id_of_matiere(char matiere[50], struct matiere table_matiere[6])` prenant en arguments une chaîne de caractères et un tableau de 6 matières et retournant l'identifiant de la matière. Par exemple `id_of_matiere("Physique", table_matiere_exemple)` devra retourner `2`.

Q. 5 Définir une fonction `valide` prenant en arguments un étudiant, la table des matières et le nom d'une matière et testant si l'étudiant valide la matière.

Q. 6 Définir une fonction `moyenne_matiere` prenant en argument le nom d'une matière, un tableau de matières, un tableau d'étudiants (et sa longueur) et calculant la moyenne (un flottant) des étudiants dans cette matière.

Q. 7 Définir une fonction `moyenne_etu` prenant en arguments un numéro étudiant, un tableau d'étudiants (et sa longueur) et calculant la moyenne de l'étudiant en question.

Q. 8 Définir une fonction `imprime_moyennes` prenant en argument un tableau d'étudiants (et sa longueur) et imprimant l'ensemble des moyennes sous la forme suivante :

```
Lovelace, Ada a une moyenne de 10.166667
Turin, Alan a une moyenne de 10.666667
Hamilton, Margaret a une moyenne de 12.666667
Church, Alonzo a une moyenne de 12.416667
Cousot, Radhia a une moyenne de 16.666666
```

Q. 9 Définir une fonction `imprime_meilleurs` prenant en arguments un tableau de matières, un tableau d'étudiants (et sa longueur) et imprimant les étudiants ayant eu les meilleurs notes dans chaque matière sous la forme suivante :

```
La meilleure note en Anglais est 17.000000, obtenue par Cousot Radhia
La meilleure note en Mathématiques est 19.000000, obtenue par Cousot Radhia
La meilleure note en Informatique est 18.000000, obtenue par Hamilton Margaret
La meilleure note en Physique est 18.000000, obtenue par Cousot Radhia
La meilleure note en Français est 16.500000, obtenue par Church Alonzo
La meilleure note en SI est 17.000000, obtenue par Hamilton Margaret
```

Q. 10 Définir un type structuré `matiere_note` représentant une matière et contenant non seulement le nom, l'identifiant et le seuil de validation mais aussi la moyenne des notes obtenues par les étudiants.

Q. 11 Définir une fonction `complete_moyennes` prenant en arguments un tableau de matières, un tableau d'étudiants (et sa longueur), un tableau de matières avec notes et remplissant le tableau de matières avec notes pour y faire figurer les moyennes obtenues par les étudiants dans chaque matière.

Exercice 3 : Inventaire

Dans cet exercice on s'intéresse à la gestion d'un inventaire d'un magasin. On fixe pour tout l'exercice un nombre (100) maximal d'éléments dans l'inventaire du magasin. Un *objet* de l'inventaire est déterminé par 3 grandeurs : un identifiant unique (une chaîne de 6 caractères), son prix (un flottant), la quantité encore en réserve.

Q. 1 Définir un type structuré `objet` contenant ces 3 grandeurs.

Le stock est déterminé par 2 grandeurs : un tableau de 100 objets et un entier n représentant le nombre d'objets différents stockés dans le tableau (les objets seront alors stockés dans les n premières cases du tableau).

Q. 2 Définir un type structuré `stock` contenant ces 2 grandeurs.

Q. 3 Définir une fonction `nouveau_objet` prenant en arguments **un pointeur** vers le stock courant du magasin, un identifiant unique (une chaîne de 6 caractères), un prix et **modifiant** le stock courant pour y ajouter un nouvel objet ayant cet identifiant, ce prix et une quantité en réserve nulle (s'il existe déjà un objet ayant cet identifiant dans le stock, il restera inchangé).

Q. 4 Définir une fonction `ajout_objet` prenant en arguments **un pointeur** vers le stock courant du magasin, un identifiant unique, et un entier x, et ajoutant x à la quantité en réserve de l'objet ayant cet identifiant.

Q. 5 Définir une fonction `soldes` prenant un arguments **un pointeur** vers le stock courant du magasin, un caractère c, et un entier n et appliquant sur tous les objets du stock ayant un identifiant unique dont la première lettre est la caractère c une réduction de $n\%$.

Q. 6 Définir une fonction `affiche_stock` prenant un argument **un pointeur** vers le stock courant du magasin et affichant tous les objets du stock (pour chaque objet on affichera, son identifiant, son prix et la quantité en réserve).

Q. 7 Définir une fonction `affiche_rupture` prenant un argument **un pointeur** vers le stock courant du magasin et affichant l'identifiant de tous les éléments en rupture de stock.

On définit un achat comme étant la donnée d'un identifiant unique (représentant un objet) et d'une quantité à acheter.

Q. 8 Définir un type structuré `achat` contenant ces 3 grandeurs.

Q. 9 Définir une fonction `achete` prenant en arguments **un pointeur** vers le stock courant du magasin, un tableau d'achats (et sa longueur), et retournant la somme des valeurs de tous les achats. On modifiera le nombre d'éléments en réserve pour tenir compte de ces achats.

Q. 10 Pour aller plus loin. Définir une interface sous forme de menu permettant à un utilisateur d'appliquer toutes les fonctionnalités ci-dessus sur le stock courant. Ci-dessous un exemple d'échange entre votre programme et les réponses d'un utilisateur (indiquées par du rouge).

```
Que voulez-vous faire :
0 - afficher le stock
1 - afficher les objets en rupture de stock
2 - ajouter un nouveau type d'objet au stock
3 - ajouter des objets au stock
4 - solder un type d'objets
5 - enregistrer un achat
>>> 0
```

```
Identifiant   Prix     Quantité
aexhuj        12.50    5
djcuzk        5.50     6
jhxscq        1.23     1
jxqshc        6.80     12
kaoxlq        5.00     0

Que voulez-vous faire :
0 - afficher le stock
…
5 - enregistrer un achat
>>> 2
Identifiant à ajouter :
>>> ahduje
Prix :
>>> 18.50

Que voulez-vous faire :
0 - afficher le stock
…
5 - enregistrer un achat
>>> 0
Identifiant   Prix     Quantité
aexhuj        12.50    5
djcuzk        5.50     6
jhxscq        1.23     1
jxqshc        6.80     12
kaoxlq        5.00     0
ahduje        18.50    0
```