Examen final - LO01 A2017

2 H - Sans documents (sauf diagrammes de C) ni calculette
N'oubliez pas de commenter vos programmes. Les types des paramètres et des fonctions sont parfois volontairement omis dans l'énoncé. A vous de les définir.

Vous aurez besoin de 4 copies : une par exercice

1. Calcul de l'IFI (6 points) -> copie n°1

La fiscalité change au premier janvier 2018, l'ISF est supprimé au profit de l'IFI (Impôt sur la Fortune Immobilière). Dans le fichier "patrimoine.dat" à Bercy sont stockées les informations sur le patrimoine d'un ensemble de personnes. Chaque enregistrement contient les informations suivantes :

un contribuable :
- nom (chaîne de caractères de 30 caractères maximum)
- prénom (chaîne de caractères de 30 caractères maximum)
- biens (tableau d'au maximum 200 biens)
- nbPat (nombre d'éléments contenus dans le tableau biens)

un bien :
- nom (chaîne de caractères de 30 caractères maximum)
- montant (montant entier en euros du bien)
- nature du bien (1 : immobilier, 0 : non immobilier)

1.1 / Définir le type *unBien* et *unContribuable* basé sur les informations ci-dessus.

1.2 / Ecrire la fonction *calculPatrimoineImmo(unContribuable)* qui a comme paramètre une variable de type *unContribuable* et qui renvoie la somme des montants des biens immobiliers.

1.3 / Ecrire la fonction *ajouteUnContribuable()* qui ajoute en fin de fichier *"patrimoine.dat"*, un nouveau contribuable. Les champs correspondants seront remplis par un dialogue avec l'utilisateur.

1.4 / Ecrire le programme principal, qui ajoute un contribuable au fichier *"patrimoine.dat"* et calcul le nombre de personnes qui seront soumises à l'ISI (une personne est soumise à l'IFI si elle possède un patrimoine immobilier supérieur à 1300000€).

Remarque : on considérera que l'utilisateur entre des valeurs correctes.

2. Divers (4 points) -> copie n°2

Récursivité :

2.1 / Ecrire la fonction récursive *int suiteRecursive(int n)* qui affiche la suite de nombres n n-1 n-2 … 3 2 1 et renvoie la somme de ces nombres.

2.2 / Ecrire le programme principal qui demande à un utilisateur un nombre entier, affiche la suite correspondant au nombre ainsi que la somme des valeurs affichées.

Exemple d'exécution :
```
Quelle valeur pour la suite (>0) ?  : 5
5      4      3      2      1
Somme des termes de la suite = 15
```

Appel de fonctions: Soit la fonction *void modifieVal (float *val)* qui modifie la valeur pointée par *val* .
2.3 / Ré-écrire les 4 lignes incomplètes en utilisant la variable v pour les deux premières et la variable w pour les deux suivantes.

```
int main() {
   float v=5.3,*w;

   w=&v ;

   modifieVal(...);   // utilise v
   printf("la valeur modifiée est %f", ...); // utilise v
   modifieVal(...);    // utilise w
   printf("la valeur modifiée est %f", ...) // utilise w

   return 0;
}
```

3. Intersection de deux chaînes de caractères (4 points) -> copie n°2

On veut écrire une fonction de prototype
void *intersection(char* source1, char* source2, char* dest, Rapport* r)* ;
qui calcule le résultat de l'intersection de deux chaînes de caractères *source1* et *source2*. L'intersection est définie comme étant la plus grande chaine de caractères qui est à la fois la fin de *source1* et le début de *source2*. Le résultat de l'intersection est stocké dans la chaîne de caractères pointée par *dest*.
Rapport est un type structure comprenant deux champs de type int nommé *debut* et *longueur*. La fonction intersection modifie la donnée de type *Rapport*, dont l'adresse *r* a été passée en argument, de manière à ce que son champ *debut* soit égale au numéro du premier caractère de *source1* qui marque l'intersection, et où son champ *longueur* soit égale à la longueur de la chaine *dest*. Si l'intersection est vide, *debut* est égale à la longueur de *source1* et longueur est égale à 0.

Exemple :
```
#define MAX 100

int main() {
        char res[MAX], Rapport r ;
        intersection("poisson", "sonate", res, &r);
        printf("%s %d %d\n", res,r.debut,r.longueur);  // affiche son 4 3
        return 0;
}
```

3.1 / Définir la structure Rapport.

3.2 / Décrire en quelques lignes le fonctionnement d'un algorithme permettant de calculer l'intersection de deux chaines de caractères.

3.3 / Définir la fonction *intersection*.

4. Mastermind (6 points) -> copie n°4

Le Mastermind est un jeu où l'utilisateur doit deviner une combinaison de 4 couleurs (6 couleurs différentes possibles). Ici, l'ordinateur choisira la combinaison de manière aléatoire.

A chaque tour, l'utilisateur propose 4 couleurs, et en retour les informations suivantes lui sont communiquées:
- le nombre de couleurs bien placées (la couleur existe dans la combinaison et est à la bonne position)
- le nombre de couleurs mal placées (la couleur existe dans la combinaison mais n'est pas à la même position).

Pour simplifier l'affichage et les saisies utilisateur, les couleurs seront représentées par des entiers de 1 à 6.

4.1 /Ecrire une fonction *initCombinaison()* qui remplira, de manière aléatoire, un tableau "combinaison" passé en paramètre avec 4 couleurs différentes choisies parmi les 6 possibles.

4.2 /Ecrire une fonction *lectureProposition()*, qui prendra en paramètre le tirage aléatoire généré par l'ordinateur et qui le comparera avec une proposition saisie par l'utilisateur dans la fonction (4 couleurs à saisir).

La fonction affichera le nombre de couleurs bien placées et mal placées.
La fonction renverra 1 si la proposition est correcte (4 couleurs bien placées), 0 sinon.

4.3 /Ecrire la fonction principale qui initialisera un tirage aléatoire et analysera les propositions de l'utilisateur.

Le jeu s'arrête lorsque la combinaison est trouvée ou lorsque le joueur a épuisé son nombre d'essais (12 essais maximum).

Exemple de partie:
L'ordinateur choisit la combinaison 5 4 6 2
```
Entrez les 4 couleurs: 1 2 3 4   -> couleurs bien placées: 0, mal placées: 2
Entrez les 4 couleurs: 1 2 4 5   -> couleurs bien placées: 0, mal placées: 3
Entrez les 4 couleurs: 5 4 6 1   -> couleurs bien placées: 3, mal placées: 0
Entrez les 4 couleurs: 5 4 2 1   -> couleurs bien placées: 2, mal placées: 1
Entrez les 4 couleurs: 5 4 6 2   -> couleurs bien placées: 4, mal placées: 0
Trouvé en 5 coups
```