Devoir surveillé de C, 13 mai 2014

Tous documents interdits à l'exception d'une page A4 recto, manuscrite de votre main. La correction tiendra compte de la qualité de l'argumentaire et de la présentation. Le barème indiqué est approximatif.

Exercice 1 : Compter les voyelles (chaînes de caractères et fichiers – 4pts)

On souhaite écrire un programme affichant le décompte des voyelles présentes dans une chaîne de caractères donnée.

(1.1) Écrivez un programme complet prenant une chaîne de caractères à analyser en argument de la ligne de commande, de la manière suivante (2pts).

```
$ ./compte-voyelles "Bonjour. Bon courage pour ce partiel."
Nombre de a: 2
Nombre de e: 3
Nombre de i: 1
Nombre de o: 5
Nombre de u: 3
Nombre de y: 0
```

(1.2) Modifiez votre programme afin qu'il lise le texte à analyser depuis un fichier dont le nom est passé en argument de la ligne de commandes (2pt).

Exercice 2 : Gestion de parc locatif (programmation structurée, POO – 6pts)

Pour gagner en efficience, un loueur de voitures décide de concevoir son propre progiciel de gestion intégré (ERP) pour gérer son parc. Chaque voiture est définie par sa catégorie (Citadine, Berline, Familiale, Utilitaire), sa plaque d'immatriculation et son prix par jour. Chaque voiture peut également être actuellement disponible, ou bien déjà louée.

(2.1) Écrivez une structure de données `voiture_t` contenant toutes les informations relatives à une voiture donnée. Vous suivrez l'approche plutôt orientée objet vue en cours. (½pt)

(2.2) Écrivez une fonction `voiture_new()`, prenant en arguments les informations initiales, et retournant une référence à un «objet», c'est-à-dire un pointeur sur une nouvelle structure `voiture_t` (1pt).

(2.3) Écrivez une fonction `voiture_free()`, prenant une référence à une voiture pour la détruire en libérant la mémoire qu'elle occupait (½pt).

(2.4) Écrivez une fonction `voiture_est_libre()` retournant si oui ou non la voiture en paramètre est libre. Écrivez une fonction `voiture_loue()` qui s'assure que la voiture en paramètre n'est plus libre (½pt).

(2.5) Écrivez la déclaration d'un tableau nommé `voitures`, permettant de stocker les informations relatives à toutes les voitures du loueur. Une variable `nb_voitures` stockera la taille de ce tableau (½pt).

(2.6) Écrivez une fonction `voiture_achete()` créant une nouvelle voiture grâce à `voiture_new()` ci-dessus et qui l'ajoute au tableau. Comment être sûr que le tableau est assez grand ? (2pt)

(2.7) Écrivez une fonction qui étant donné une catégorie de voitures, retourne un pointeur vers la première voiture de cette catégorie qui ne soit pas encore louée, ou NULL si aucune n'est disponible (1pt).

Exercice 3 : Questions de cours (4pts)

(3.1) Que signifient les messages d'erreur suivants ? Quand arrivent-ils, et comment les corriger ? (2pts).

(a) *Control reaches end of non void function*

(b) *Implicit declaration of function toto*

(3.2) Expliquez les problèmes posés par ces morceaux de programmes, puis proposez des solutions (2pts).

Programme 3.2a

```
1 char *truc;
2 *truc = 'x';
```

Programme 3.2b

```
1 char *truc = "constant";
2 truc[0] = 'x';
3 free(truc);
```

Programme 3.2c

```
1 int *make_buff(int a) {
2   int buff[SIZE], cpt;
3   for (cpt=0; cpt<SIZE; cpt++)
4     buff[cpt] = a;
5   return buff;
6 }
```

Exercice 4 : Lecture de code (6pts)

Indiquez ce qu'affiche l'exécution des programmes C suivants, qui compilent et s'exécutent sans erreur.

Programme 4.1 (début)

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  void f1 ( int a, int *b ) {
4    a = *b;
5  }
6  void f2 ( int *b, int c ) {
7    *b = c;
8  }
9  void f3 ( int *a, int c ) {
10   f4 ( &a, c );
11 }
12 void f4 ( int **b, int a ) {
13   **b = a;
14 }
```

Programme 4.1 (fin)

```
15 int main () {
16   int x = 5, y = 7, z = 9;
17   f1 ( x, &y );
18   printf ( "x = %d, y = %d, z = %d\n", x, y, z );
19   f2 ( &x, y );
20   printf ( "x = %d, y = %d, z = %d\n", x, y, z );
21   f3 ( &y, z );
22   printf ( "x = %d, y = %d, z = %d\n", x, y, z );
23   return 0;
24 }
```

Programme 4.2

```
1  #include <stdio.h>
2  int main(int argc, char **argv) {
3
4    int a[] = {12, 23, 34, 45, 56, 67, 78, 89, 90, 100};
5    int *p = a;
6
7    printf("a: %d\n", *p+2);
8    printf("b: %d\n", *(p+2));
9    printf("c: %d\n", &p+1);
10   printf("d: %d\n", a+3);
11   printf("e: %d\n", &a[7]-p);
12   printf("f: %d\n", p+(*p-10));
13 }
```