Licence Sciences, Technologies, Santé
Semestre 3 (L2) - Parcours Mathématiques, Informatique

Université de Perpignan Via Domitia
Année universitaire 2019/2020

Contrôle terminal de Programmation en C avancée

Mardi 7 janvier 2020 de 14h à 16h

Aucun document n'est autorisé.
Les calculatrices, ordinateurs, et téléphones portables sont interdits.

Exercice 1. Quelle note voulez-vous ? — $1 + 0.5 = 1.5$ *pts*

Pour information, votre note de Programmation C sera calculée de la manière suivante :

$$\text{note} = 0.5 \times \text{CT} + 0.5 \times \max(\text{CC1}, \text{CC2}).$$

► **1.** Écrire un programme qui lit vos notes de CT, CC1 et CC2, puis qui calcule et affiche votre note finale.

► **2.** Modifier le programme pour afficher si vous validez la matière (note supérieure ou égale à 10).

Vous pouvez écrire un seul programme en indiquant quelles parties correspondent aux réponses 1 et 2, respectivement.

Exercice 2. Deux boucles à un point — $1 + 1 = 2$ *pts*

► **1.** Écrire un programme qui lit une valeur entière $n \geq 1$, puis qui affiche la suite de chiffres suivante, en utilisant une ou plusieurs boucles **`for`** :

$$(1)(12)(123)(1234)(12345)(123456)\cdots(12\cdots n).$$

► **2.** Modifier le programme (*en ne donnant que la partie qui diffère de la question précédente*) pour utiliser, cette fois-ci, une ou plusieurs boucles **`while`**.

Exercice 3. Développement limité — $1.5 + 1 + 1.5 = 4$ *pts*

Le développement limité en 0 de $\cos(x)$ est définit de la manière suivante :

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots + (-1)^n \frac{x^{2n}}{(2 \cdot n)!}.$$

► **1.** Écrire une fonction `factorielle` qui, étant donnés deux entiers représentant $k \geq 0$ et $(2 \cdot k)!$, retourne la valeur de $(2 \cdot (k+1))!$ calculée de la manière suivante : $(2 \cdot (k+1))! = (2 \cdot k)! \times (2 \cdot k + 1) \times (2 \cdot k + 2)$.

► **2.** Sur le même modèle, écrire une fonction `pow_x` qui, étant donnés deux réels représentant x et x^{2k}, calcule et retourne $x^{2(k+1)}$.

► **3.** Écrire enfin un programme qui lit au clavier une valeur réelle $x \in [-1, 1]$ et une valeur entière $n \geq 1$, puis qui affiche la valeur de $\cos(x)$ obtenue en utilisant la formule précédente.

Exercice 4. À l'aide ! — 2 *pts*

Un étudiant vous envoie le mail suivant : "*Je ne comprends pas, mon programme C compile, mais il ne fait pas ce que je veux. Normalement, je devrais obtenir "S= 35 et P= 306", et ce n'est pas le cas. Mais je vous assure, il n'y a pas d'erreur !*". Pour l'aider, vous lui demandez de vous envoyer son programme (ci-dessous).

```
1   void somme_produit(int a, int b,
2                      int S, int P) {
3     S = a + b;
4     P = a * b;
5   }
6
```

```
7   int main(void) {
8     int a = 17, b = 18, S = 0, P = 0;
9     somme_produit(a, b, S, P);
10    printf("S= %d et P= %d \n", S, P);
11    return 0;
12  }
```

► **1.** L'étudiant a évidemment tort. Corriger le programme en expliquant précisément chaque erreur (*en vous aidant éventuellement des numéros de ligne*). Quelles notions vue en cours sont illustrées dans cet exemple ?

Exercice 5. Tri pour deux valeurs

$1.5 + 0.75 + 0.75 + 1 = 4$ *pts*

Soit t un tableau de taille n, où chaque élément ne peut être que 0 ou 1. On souhaite trier le tableau par ordre croissant. Par exemple, pour le tableau $t = \{0, 1, 1, 0, 0\}$ de taille $n = 5$, le résultat sera $t = \{0, 0, 0, 1, 1\}$.

► **1.** Écrire une fonction qui, étant donné un tableau t de taille 10, trie ce tableau en utilisant une méthode par comptage.

Indice : une méthode par comptage consiste à déterminer le nombre d'apparitions de chaque valeur (0 et 1), puis de construire le tableau résultat en fonction.

► **2.** Écrire un programme principal, qui permet d'illustrer son utilisation sur le tableau $t = \{0, 1, 1, 0, 1, 0, 0, 1, 1, 0\}$.

► **3.** Donner le prototype de la fonction qui permet de faire le même traitement sur un tableau de taille n.

► **4.** Proposer maintenant une nouvelle fonction, qui trie le tableau en ne le parcourant qu'une seule fois (en utilisant une méthode proche du *tri sélection-permutation)*.

Exercice 6. Les pointeurs, comment ça marche ?

$1 + 1.5 = 2.5$ *pts*

► **1.** Rappeler la définition d'un pointeur vue en cours, en illustrant éventuellement avec un schéma.

► **2.** Quel est le résultat de l'exécution du programme suivant. Donner une explication (*en vous aidant éventuellement des numéros de ligne et/ou d'un schéma)*.

```
1   int
2   main(void) {
3     int a = 2, b = a;
4     int * ptr = &a;
5     *ptr = a * (*ptr);
6     printf("--> a= %d et *ptr= %d... et b= %d\n", a, *ptr, b);
7     a = ((*ptr)--) - 1;
8     printf("--> a= %d et *ptr= %d... et b= %d\n", a, *ptr, b);
9     a = (*ptr--) - 1;
10    printf("--> a= %d et *ptr= %d... et b= %d\n", a, *ptr, b);
11    return 0;
12  }
```

Exercice 7. Points les plus proches dans un nuage

$1 + 2 = 3$ *pts*

La distance d entre deux points A et B, de coordonnées (x_A, y_A) et (x_B, y_B) dans le plan, respectivement, est donnée par la formule suivante :

$$d = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}.$$

► **1.** Écrire une fonction `distance` qui, étant données les coordonnées de deux points, calcule et retourne la distance entre ces deux points dans le plan.

Maintenant, étant donné un nuage de points de dimension $n \geq 2$, on souhaite déterminer les deux points les plus proches dans le plan. Pour chaque point, on connaît les coordonnées (x, y).

► **2.** Proposer une manière de représenter le nuage de points. Écrire ensuite une fonction qui, étant donnés un nuage de points et sa dimension n, détermine et retourne l'indice des deux points les plus proches, en faisant au plus $n \cdot (n-1)/2$ appels à la fonction `distance`.

Exercice 8. Matrice symétrique

$2 + 1 = 3$ *pts*

Soit M une matrice carrée de taille $n \times n$ d'entiers. Une matrice carrée est symétrique si elle est égale à sa propre *transposée*. L'exemple ci-dessous donne une matrice carrée et sa *transposée* :

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \xrightarrow{\text{transposée}} \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}.$$

On représentera une matrice par un tableau 2D de taille $n \times n$.

► **1.** Écrire une fonction qui détermine la transposée d'une matrice carrée de taille $n \times n$ passée en paramètre.

► **2.** Écrire finalement une fonction qui retourne un entier indiquant si la matrice M passée en paramètre est symétrique (retour $= 1$) ou non (retour $= 0$).