ULP/Strasbourg I Licence de sciences $1^{ière}$ année : mentions PA/PCM

Introduction à la programmation scientifique
Responsables : Christian Boily

Examen de contrôle final
Session de juin 2008
Durée : 1 heure

Appel lundi le 26 mai 2008 à 10h15
Amphithéâtres 5 & 6, Bâtiment Atrium Campus de l'Esplanade

Les notes personnelles, soit les notes de cours, TD et TP, sont permises. Aucun livre de référence ou support électronique (téléphone cellulaire, calculette, etc) n'est autorisé, à l'exception des dictionnaires de langues qui devront être déclarés au début de l'épreuve. Un maximum de 20 points sera attribué.

Exercice 1 (7 points)

Indiquez par une croix (X) dans le tableau ci-bas tous les types possibles pour les variables sujettes aux affectations suivantes :

```
y = 2 / 3.0 ; x = sin( x*y ) ;  z.re = (float)2 / 3 ;
z.tx = "Ceci est difficile" ; c = 'C' ; *p= y+1.2;
```

Les déclarations sont valables pour l'exercice en son entier. Il est demandé de respecter le type du résultat de chaque opération arithmétique.

variable	char	int	unsigned int	float	double	struct	pointeur
y							
x							
z							
z.re							
z.tx							
c							
p							

Exercice 2 (7 points)

Écrivez un programme C permettant à l'utilisateur de deviner si deux nombres entiers sont multiples l'un de l'autre.

- le code devra saisir les deux nombres à la ligne de commandes
- le code renverra la réponse oui/non selon que le résultat est positif ou négatif
- Dans le cas positif, en plus de la réponse, la solution sera donnée explicitement. Donc par exemple si a et b sont les nombres saisis, "Le nombre a = 2 fois le nombre b" sera affiché pour le cas où a = 2b.

Exercice 3 (7 points)

Cet exercice se base sur le programme suivant :

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 int nouvel_e( int i, int j, float aa[] ) {
4 aa[i]=aa[i] + aa[j] ; aa[j] = aa[i] - aa[j] ; aa[i] = aa[i] - aa[j] ;
5  return ; }
6   int main() { int n,i=0, j=1;
7   printf( "Dimension de la table ? " ) ;
8   scanf( "%d, &n );
9   float table[n] ;
10   for( ; i < n ; i++ ) {
11     printf( "\n pour i = %d, donner la valeur : ", i );
12     scanf( "%f", &table[i] ) ; }
13 for( i=0; i < n ; i=i+1 ) {
14    for( j = i+1 ; j < n ; j++ ) {
15      if( table[i] < table[j] ) {  ;}
16      else { nouvel_e(i,j, table) ; }
17 } }
18 printf( " { " ) ;
19 i = 0 ; while( i < n ) { printf( " %3.1f,", table[i] ) ;i++ ;}
20 printf( " } \n\n" ) ;
/* fin du programme */
21 return 0;
22 }
```

1) Modifiez le programme en éliminant l'une des boucles *for* (ligne 13) pour que n'apparaissent plus que deux boucles.

2) Expliquez ce que fait la fonction *nouvel_e*. Récrivez la fonction en ne faisant intervenir aucune opération arithmétique (+,-,* et /) en faisant intervenir une variable tempon (temporaire).

Exercice 4 (7 pts)

On cherche à coder une fonction qui permet de reproduire le OU EXCLUSIF (c'est-à-dire XOR) de la table de logique booléenne : c'est-à-dire

si (a,b) = (1,0) ou (0,1), alors
XOR = 1 ;
sinon XOR = 0.

On veut développer une fonction XOR prenant deux arguments entiers dont les valeurs peuvent valoir seulement 0 ou 1. Codez cette fonction en C de manière à ce qu'elle renvoie 1 dans le cas où un seul des arguments vaut 1, et 0 dans *tous* les autres cas.