Automne 2020 **Examen médian- LO01**

Vous aurez besoin de 3 copies : une par exercice

2 H - Sans documents (sauf diagrammes de C) et une feuille resto-verso avec notes manuscrites

La clarté de vos réponses sera prise en compte. Ne pas écrire au crayon.

Les types des paramètres dans les entêtes des fonctions sont volontairement omis. À vous de les compléter

N'oubliez pas de commenter vos programmes.

1. Calcul, pointeurs et entiers (7 points) Copie 1

1.1/ Calculer, en indiquant comment elles sont évaluées, les expressions suivantes :

```
10 + 59 % 5
5.0 / 2.0 != 5 / 2
'9' - '8' == 1
-3 <= -2 <= -1
8 < 9 && 'a' == 'A'
x = 8
```

1.2/ Qu'affiche le printf ?

```
float x , y;
float *pt1, *pt2;

pt1 = &x ;
*pt1 = 5.0 ;
y = x + 3.0;
pt2 = pt1;
*pt2 = *pt1 * 2.0;
pt1= &y;
(*pt1)-- ;
printf(" x= %f, y= %f, *pt1= %f, *pt2= %f\n", x, y, *pt1, *pt2);
```

1.3/ Écrire un programme qui lit une suite de caractères terminée par un # et détermine si celle-ci correspond à un nombre entier ou réel, positif ou négatif. Si ce n'est pas le cas, le programme indiquera "Ce n'est pas un nombre".

- Un entier est composé d'une suite de chiffres, précédée du signe – (moins) pour les nombres négatifs.
- Un réel est composé d'une suite de chiffre avec un point pour séparer la partie décimale de la partie entière, précédée d'un signe – (moins) pour les nombres négatifs.

Exemples :

```
entrez un nombre entier ou réel terminé par un #:    -1.23#
c'est un nombre réel négatif
entrez un nombre entier ou réel terminé par un #:    4567z#
Ce n'est pas un nombre
entrez un nombre entier ou réel terminé par un #:    7890#
C'est un nombre entier positif
```

Remarque importante : cet exercice doit se faire sans utiliser de chaîne de caractères ni de tableau

Corrigé exercices 1 (1,5 + 1 + 3 points)

1)

```
10 + (59 % 5)      → 10 + 4 → 14
(5.0 / 2.0) != (5 / 2)   → 2.5 != 2 → 1 (vrai)
('9' - '8') == 1 → 1 == 1 → 1 (vrai)
(-3 <= -2) <= -1     →  1 <= -1 → 0 (faux)
(8 < 9) && ('a' == 'A')   → 1 && 0   → 0 (faux)
x = 8 → 8 (le résultat d'une affectation est la valeur affectée)
```

2) Qu'affiche le `printf` ?

```
x= 10.000000, y= 7.000000, pt1= 7.000000, pt2= 10.000000
```

3)

```c
#include <stdio.h>

int main()
{
   char carlu;  //caractère courant
   int signe = 0;  // indique si il le nombre commence ou non par le signe -
   int point = 0;  // calcule le nombre de points dans la suite de caractère
   int nombre=1;   // indique si le format est conforme à un nombre. Est mis à 0 dès qu'on detecte un caractère different du point ou d'un chiffre.

   printf("entrez un nombre entier ou réel terminé par un #:");
   carlu = getchar();
   /* teste si le premier caractère est le signe - */
   if (carlu == '-') {
      signe = 1;
      carlu = getchar();
   }
   /* boucles sur les caractères suivants : chiffres ou . */
   while (carlu!='#') {
      if (carlu == '.')
         point++;
      else
         if (carlu<'0' || carlu >'9')
            nombre=0;
      carlu = getchar();
   }

   /* sortie de boucle :  affichage du résultat en fonctions des différentes valeurs */
   if (nombre == 0 || point>=2)
      printf ("ce n'est pas un nombre\n");
   else {
      printf("c'est un nombre");
      if (point == 0 ) printf(" entier ");
      else  printf(" réel");
      if (signe == 0) printf(" positif\n");
      else printf(" négatif\n");
   }

   return 0;
}
```

2. Nombres aléatoires (7 points) Copie 2

On veut écrire un programme permettant d'afficher un ensemble de nombres aléatoires compris entre 1 et 9999. Pour cela, on demande à l'utilisateur de rentrer le nombre de lignes et de colonnes sachant que ceux-ci devront être compris entre 4 et 10 (on s'assurera que l'utilisateur a bien entré deux nombres corrects). Puis pour chaque ligne vous afficherez la somme des nombres. Pour obtenir un affichage aligné (tel que l'exemple ci-dessous), le nombre d'étoiles entre chaque nombre dépend du nombre de caractères de chaque nombre.

Question.

Écrire le programme permettant de générer l'affichage suivant :

Entrez le nombre de lignes compris entre 4 et 10 : 10
Entrez le nombre de colonnes compris entre 4 et 10 : 10

9740*4959*4225*1101*1069*2958*7533*4738*1997*4984*-> somme : 43304
9481*3227*7016*2684*9713*257**6563*2791*3284*7062*-> somme : 52078
8599*713**5650*487**5282*6962*3704*7376*5018*8274*-> somme : 52065
9226*7674*6123*2011*5753*275**1730*2130*7338*597**-> somme : 42857
3888*7559*1521*1118*8834*5598*3035*7307*4566*6065*-> somme : 49491
784**9930*7169*9184*4655*9810*1475*2555*1021*263**-> somme : 46846
426**847**5032*4991*3987*1600*7276*5584*2207*3778*-> somme : 35728
8824*7503*2262*6292*7607*4197*5256*3140*3730*578**-> somme : 49389
6528*2359*7310*410**537**6514*7728*2642*1****7034*-> somme : 41063
3622*5954*1899*534**5709*9075*2945*2422*8134*1675*-> somme : 41969

Dans le cas d'un nombre généré comportant 4 chiffres, une seule étoile le sépare du nombre suivant. Dans le cas d'un nombre généré comportant un seul chiffre, le nombre d'étoiles le séparant du nombre suivant est de 4.

Corrigé exercice 2

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MIN 4
#define MAX 10
#define NBMAX 9999

int main()
{
int nb ;                // Nombre généré aléatoirement
int nbL,nbC ;           // Nombre de lignes et nombre de colonnes entrés par l'utilisateur
int nbChiffres;         // Nombre de chiffres composant nb
int i,j,k ;             // Compteurs pour les boucles
int som;                // Somme des nombres de chaque ligne

// Saisie du nombre de lignes avec test par rapport au minimum et maximum
do{
   printf("Entrez le nombre de lignes compris entre %d et %d : ",MIN,MAX);
   scanf("%d",&nbL);
}while (nbL<MIN || nbL>MAX);

// Saisie du nombre de colonnes avec test par rapport au minimum et maximum
do{
   printf("Entrez le nombre de colonnes compris entre %d et %d : ",MIN,MAX);
   scanf("%d",&nbC);

}while (nbC<MIN || nbC>MAX);

srand(time(NULL));      // Initialisation du nombre de départ pour la génération de la suite aléatoire
printf("\n");

for(i=1; i<=nbL; i++){          // Boucle sur le nombre de lignes
   som = 0;                     // Initialisation de la somme à chaque début de ligne
   for(j=1; j<=nbC; j++){       // Boucle sur le nombre de colonnes

      nb=rand()%NBMAX + 1;      // Génération du nombre aléatoire entre 1 et 9999 compris
      printf("%d",nb);          // Affichage du nombre

      som += nb;                // Addition du nombre à la somme

      // Calcul du nombre de chiffres composant le nombre nb
      nbChiffres = 1;
      while(nb>9){
         nbChiffres++;
         nb/=10;
      }

      // Affichage du nombre d'étoiles en fonction du nombre de chiffres du nombre nb
      for(k=1; k<=4-nbChiffres; k++)
         printf("*");
      printf("*");
   }
   printf("-> somme : %d \n",som);      // Affichage de la somme à la fin de la ligne et retour à la ligne
}

return 1;

}
```

3. Mesurer le désaccord par rapport à ses préférences (7 points) Copie 3

La distance tau de Kendall est une métrique utilisée pour mesurer le « désaccord » d'une séquence d'éléments par rapport à un ordre de préférence entre ces éléments.
Dans la suite, nous nous intéresserons uniquement à des séquences ne comprenant que des chiffres entre 1 et 9. Une séquence sera représentée par un entier *long* positif. Pour cela, nous utiliserons le type *entierP* qui permet de représenter n'importe quel entier positif comportant jusque 19 chiffres.
Par exemple :
entierP seq=3567112332411982; */* est une séquence de chiffres représentée par un long entier */*
Avec des variables de type *entierP*, il est possible de faire les mêmes opérations (addition, multiplication, division, modulo) qu'avec une variable de type *int*. L'instruction de formatage pour afficher et saisir de telles valeurs avec printf et scanf est *« %llu »*.
Soit une séquence *pref* représentant une permutation des chiffres *{1, 2, 3, 4, 5, 6, 7, 8, 9}*, c'est-à-dire que *pref* est un nombre comprenant une et une seule fois chaque chiffre dans un certain ordre. Une telle permutation représente un ordre de préférence entre les chiffres. Si le chiffre *i* est située avant le chiffre *j* dans *pref*, on dit que *i* est préféré à *j*. On considère qu'un chiffre *i* n'est jamais préféré à lui-même.
Question.
Écrire un programme qui demande à l'utilisateur de saisir une séquence de préférence *pref* et une séquence de chiffres *seq*, et qui calcule et affiche la distance tau de Kendall de la séquence *seq* par rapport à la préférence *pref*. Il s'agit de calculer la somme des désaccords dans *seq*, c'est-à-dire le nombre de fois où un chiffre *i* est placé avant un chiffre j (différent de *i)* dans *seq* alors que *j* est préféré à *i* dans *pref*.
Exemples :

- Si pref=123 et seq=3211,
  - → **1**23 : 3211 : 3 et 2 se trouvent devant le premier 1 donc distance = 2
  - → **1**23 : 3211 : 3 et 2 se trouvent devant le deuxième 1 donc distance = 2 + 2 = 4
  - → 1**2**3 : 3211 : 3 se trouve devant le 2 donc distance = 4 + 1 = 5
  - → 12**3** : 3211 : pas de distance supplémentaire car 3 n'a aucun chiffre devant lui.
  - → La distance calculée est donc de 5.

- Si pref=123456789 et seq=493115627152, la distance calculée est 33.
- Si pref=298743156 et seq=493115627152, la distance calculée est 27.

Corrigé exercice 3 :

```
#include <stdio.h>
#include <stdlib.h>
#define entierP unsigned long long


int main()
{

entierP  pref,seq;
entierP  tmpSeq,tmpPref;

int x, y, d = 0; // distance

printf("Entrez votre préférence :");
scanf("%llu", &pref);

printf("Entrez votre séquence :");
scanf("%llu", &seq);

printf("pref=%llu, seq=%llu \n", pref, seq);

while(seq != 0) {

   x = seq % 10; // dernier chiffre
   seq /= 10; // on élimine le dernier chiffre
   tmpSeq = seq; // copie temporaire de n moins le dernier chiffre

 // analyse de tous les nombres situés avant x dans seq

   while(tmpSeq != 0) {

                    y = tmpSeq % 10; // y est un nb avant x dans seq

      // est ce que c'est un nb avant x dans pref ?

                    tmpPref = pref;

      while(tmpPref % 10 != x && tmpPref % 10 != y)

         tmpPref /= 10;

      if(x!=y && tmpPref % 10 == y) d++;  // y est situé avant x

      tmpSeq /= 10;

   }

}
printf("distance=%d\n",d);
return 0;

}
```