CSC 3002 - Enoncé du Contrôle Final 2
Vendredi 3 avril 2009

Durée : 2 heures
Document autorisé : le polycopié distribué en cours

Exercice d'algorithmique (sur 6 points)

Un nombre est dit de *Harshad* s'il est divisible par la somme de ses chiffres. *Harshad* n'est pas un patronyme ; c'est un terme qui signifie grande joie en sanskrit.
Prenons par exemple le nombre 18. La somme de ses chiffres est égale à 9 et 18 est divisible par 9.
Voici les vingt premiers nombres de *Harshad* : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 18, 20, 21, 24, 27, 30, 36, 40, 42.
Vous devez définir en **langage algorithmique** une fonction *estNombreDeHarshad* qui prend un paramètre *nb* de type naturel et qui renvoie un booléen égal à VRAI si *nb* est un nombre de *Harshad*, à FAUX sinon.
Définissez également la fonction *principale* qui permettra à l'utilisateur de saisir un naturel, appelera la fonction *estNombreDeHarshad* et affichera le résultat dans la fenêtre *shell* de manière lisible.

Exercice de langage C (sur 4 points)

Le programme ci-dessous compile et s'exécute normalement. Vous devez simplement écrire sur votre copie, ce qui s'affiche dans la fenêtre *shell* lorsqu'on l'exécute.

```
#include <stdio.h>
#include <stdlib.h>

int main ( ) {

  int m = 41, n, *p, *q, **x;

  p = &m;
  q = &n;
  *q = m / 2;
  q = p;
  printf ( "1 : m = %d, n = %d, *p = %d, *q = %d\n", m, n, *p, *q );

  x = &q;
  m = n + 1;
  **x = 17;
  *p = **x;
  *p = *q;
  printf ( "2 : m = %d, n = %d, *p = %d, *q = %d, **x = %d\n",
           m, n, *p, *q, **x );

  return EXIT_SUCCESS;
}
```

Exercice de structures de données (sur 10 points)

Considérons une suite de valeurs réelles ; cette suite, de longueur indéterminée, peux contenir la même valeur plusieurs fois. Elle est stockée sous forme de liste linéaire **non triée** gérée par pointeurs.
Voici un rappel en langage C de la définition du type *Element* adapté au stockage des valeurs de cette suite et du type *Liste* :

```
typedef struct element { double valeur;
                         struct element *lien; } Element;

typedef Element *Liste;
```

1. Vous devez définir **en langage C** une fonction *calculerMoyenneEcartType* qui permettra de déterminer la moyenne et l'ecart-type de cette suite de valeurs. Voici le prototype de cette fonction :

   ```
   void calculerMoyenneEcartType ( Liste l, double *pMoy, double *pET );
   ```

   Vous disposez de la fonction *estListeVide* (vous n'avez pas à la définir) dont voici le prototype :

   ```
   Booleen estListeVide ( Liste l );
   ```

   Cette fonction renvoie la valeur VRAI si la liste *l* est vide, FAUX sinon.
   On vous rappelle également que la bibliothèque mathématique de l'environnenemt du langage C vous fournit la fonction *sqrt* qui renvoie la racine carrée d'une valeur de type *double* passée en argument. Cette fonction est déclarée dans le fichier *math.h*.

   ```
   double sqrt(double x);
   ```

2. Supposons que cette liste contienne un grand nombre de valeurs et qu'il y ait beaucoup d'opérations d'insertion et de suppression. On voudrait pouvoir recalculer le plus rapidement possible la moyenne et l'écart-type après chaque opération d'insertion ou de suppression.
   Sans modifier le type *Element*, redéfinissez le type *Liste* comme une structure de façon à optimiser le recalcul de la moyenne et de l'écart-type.
3. Lorsqu'on crée une nouvelle liste, il faut initialiser les champs de cette structure. Définissez la fonction *nouvelleListe* dont voici le prototype :

   ```
   Liste nouvelleListe ( void );
   ```

4. Dans le cas d'une insertion d'une valeur dans la liste, il faut modifier les champs de cette structure. Définissez la fonction *inserer* dont voici le prototype :

   ```
   Liste inserer ( Liste liste, double val );
   ```

   Cette fonction doit appeler la fonction *enlister* dont voici le prototype :

   ```
   Element *enlister ( Element *ptrElement, double val );
   ```

   Vous n'avez pas à définir cette fonction.
5. Il en est de même dans le cas d'une suppression d'une valeur de la liste. Définissez la fonction *supprimer* dont voici le prototype :

   ```
   Liste supprimer ( Liste liste, double val );
   ```

   Cette fonction doit appeler la fonction *delister* dont voici le prototype :

   ```
   Element *delister ( Element *ptrElement, double val );
   ```

   Vous n'avez pas à définir cette fonction.