CSC 3002 - Enoncé du Contrôle Final 1
Vendredi 30 janvier 2009

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

Exercice d'algorithmique (sur 7 points)

Si *d* est un diviseur de *n* et s'il est premier avec [1] *n* **div** *d*, c'est un diviseur unitaire de *n*. Un nombre unitairement parfait est égal à la somme de ses diviseurs unitaires. Prenons comme exemple le nombre 60 :

diviseur de 60 (hors 60)	1	2	3	4	5	6	10	12	15	20	30
60 **div** diviseur	60	30	20	15	12	10	6	5	4	3	2
premiers entre eux ?	oui	non	oui	oui	oui	non	non	oui	oui	oui	non
diviseur retenu	1		3	4	5			12	15	20	
leur somme	60										

On ne connait que cinq nombres unitairement parfaits : 6, 60, 90, 87360 et 1,4582... 10^{23}. On conjecture qu'il sont en nombre fini.

Vous devez, **sur la feuille de réponses fournie**, définir en **langage algorithmique** une procédure *afficherNombresUnitairementParfaits* qui prend en donnée un paramètre *nb* de type naturel et qui affiche dans la fenêtre *shell* les nombres unitairement parfaits inférieurs ou égaux à *nb*.
Définissez également la fonction *principale* qui permettra à l'utilisateur de saisir un naturel et qui appelera la procédure *afficherNombresUnitairementParfaits*.

Adoptez une démarche descendante ; le problème s'y prête fort bien.

Exercice de langage C (sur 7 points)

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

expressions.c (début)

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

int main ( ) {
```

[1] a est premier avec b si et seulement si pgcd (a, b) = 1

expressions.c (fin)

```
  int t[] = { 1, 4, 7, 10, 13, 16, 19, 22, 25, 28 };
  int *p = t;

  printf ( "*p + 2 = %d\n", *p + 2 );
  printf ( "*( p + 2 ) = %d\n", *( p + 2 ) );
  printf ( "*(p + *p + 2 ) = %d\n", *(p + *p + 2 ) );
  printf ( "*(p + *( p + 2 ) ) = %d\n", *(p + *( p + 2 ) ) );
  printf ( "*(p + *( p + 2 ) - t[1] ) = %d\n", *(p + *( p + 2 ) - t[1] ) );
  printf ( "&t[7] - p + 1 = %d\n", &t[7] - p + 1 );
  printf ( "&t[7] - &( *( p + 1 ) ) = %d\n", &t[7] - &( *( p + 1 ) ) );
  printf ( "&t[7] - &( *( p + *( p + 1 ) ) ) = %d\n",
           &t[7] - &( *( p + *( p + 1 ) ) ) );

  return EXIT_SUCCESS;
}
```

Exercice de structures de données (sur 6 points)

Le nombre de Strahler a été initialement défini sur les arbres binaires par des hydrogéologues, pour décrire la forme des bassins fluviaux. Il a été ensuite utilisé pour déterminer l'espace mémoire nécessaire à l'évaluation d'une expression arithmétique présentée sous la forme d'un arbre binaire.
Soit *a* un arbre binaire, on désignera par *sag(a)* (resp. *sad(a)*) le sous-arbre gauche (resp. droit) de *a*. Le nombre de Strahler de l'arbre binaire *a* est défini par :

$$\Phi(a) = \begin{cases} 0 & si\ a\ est\ vide \\ max(\Phi(sag(a)), \Phi(sad(a))) & si\ a\ est\ non\ vide\ et\ \Phi(sag(a)) \neq \Phi(sad(a)) \\ 1 + \Phi(sag(a)) & si\ a\ est\ non\ vide\ et\ \Phi(sag(a)) = \Phi(sad(a)) \end{cases}$$

1. Inscrivez les valeur de Φ dans les carrés à coté de chaque nœud de l'arbre binaire figurant **sur la feuille de réponses fournie**.
2. Cochez **sur la feuille de réponses fournie** le parcours à utiliser pour calculer le nombre de Strahler d'un arbre binaire (il y a une seule réponse correcte).
3. Définissez **sur la feuille de réponses fournie** et **en langage C** la fonction *nombreDeStrahler* dont voici le prototype :

   ```
   int nombreDeStrahler ( Arbre a );
   ```

   On rappelle ci-dessous la définition du type *Arbre* :

   ```
   // Le type Noeud definit la representation d'un element de l'arbre
   typedef struct noeud {  int valeur;
                           struct noeud *sag;
                           struct noeud *sad; } Noeud;

   // Le type Arbre definit la representation d'un arbre binaire de recherche
   typedef Noeud *Arbre;
   ```

   Vous disposez également du type Booléen et de la fonction *estArbreVide* dont voici respectivement la définition et la déclaration :

   ```
   typedef enum { FAUX,VRAI } Booleen;

   Booleen estArbreVide ( Arbre a );
   ```