L3 Miage 2020/2021 – Évaluation Algorithmique
Durée 2h30 – aucun document autorisé
Université des Antilles

Exercice 1

Un nombre parfait est un entier égal à la somme de ses diviseurs stricts (i.e. hormis lui même).
Exemples :
6 = 1 + 2 + 3
28 = 1 + 2 + 4 + 7 + 14
Écrire un algorithme ou un programme C qui demande à l'utilisateur de rentrer un entier et qui vérifie si il est parfait ou non. Un message adéquat sera affiché dans chaque cas.

Exercice 2

A l'aide d'une ou plusieurs boucles, écrire une fonction qui prend en paramètre un tableau de quatre caractères et qui affiche tout les ensembles (hormis l'ensemble vide) que l'on peut créer à partir de ces caractères. La fonction devra renvoyer le nombre d'ensembles affichés.
L'ordre des caractères dans un ensemble n'est pas important.
Aucun autre tableau que celui reçu en paramètre ne devra être utilisé.

Exemple :
Soit un tableau contenant les caractères suivants : 'a', 'b', 'c', 'd'
la fonction affichera : (ordre des ensembles donné à titre indicatif, l'affichage peut se faire dans n'importe quel ordre)

{a}
{a, b}
{a, b, c}
{a, b, c, d}
{a, b, d}
{a, c}
{a, c, d}
{a, d}
{b}
{b, c}
{b, c, d}
{b, d}
{c}
{c, d}
{d}

et renverra 15.

Exercice 3

La formule qui permet de convertir un entier positif ou nul en base quelconque *b* est la suivante :

$$(a_n a_{n-1} \ldots a_1 a_0)_b = \sum_{i=0}^{n} a_i b^i$$

an an-1 … a1 a0 sont les chiffres du nombre en base *b.*
Exemples :
b = 2 (binaire) 53 = 1*32 + 1*16 + 1*4 + 1*1 = $1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0$ = 110101
b = 8 (octal) 53 = 6*8 + 5*1 = $6*8^1 + 5*8^0$ = 65

Pour passer en base *b*, il faut donc diviser le nombre à convertir par les puissances successives de *b*.
On pourra également utiliser l'opérateur % qui renvoie le reste d'une division entière. Pour les bases supérieures à 10, il est nécessaire de rajouter des symboles après 9. On prend donc par convention les lettres A, B, C, D, … Exemple : avec *b* = 16 (hexadécimal) 58 = 3*16 + 10*1 = $3*16^1 + 10*16^0$ = 3A

a) Écrire une fonction *puissance* qui prend en paramètres deux entiers x et y et qui renvoie x^y .

b) A l'aide de cette fonction (ne pas utiliser pow !) , écrire une fonction qui prend en paramètres deux entiers N et B (avec B inférieur ou égal à 16, hexadécimal) et qui affiche le résultat de la conversion de N en base B. On considère qu'il faut au maximum 10 chiffres en base B pour convertir N. Les chiffres non utilisés pourront être affichés à 0. Ex : 53 = 0000110101 (binaire), 53 = 0000000065 (octal) etc...

c) Écrire une fonction qui prend en paramètre un tableau de caractères, sa taille T et une base B (<=16) et qui renvoie le résultat de la conversion de la chaîne en base 10. On considère que la chaîne n'est composée que de caractères numériques et de lettres A, B, C, D, E, F. Pour passer d'un caractère numérique c ('1','2','3'…,'9') à l'entier correspondant, il faut faire un ET binaire entre c et la valeur 15. Exemple : i = c & 15 ;

Exercice 4

a) Écrire une fonction qui prend en paramètre un entier et qui renvoie 1 si celui-ci est premier et 0 sinon.

b) A l'aide de cette fonction écrire une seconde fonction prenant en paramètre deux entiers M et N avec M<N et qui affiche la liste des nombres premiers jumeaux compris entre M et N inclus.

Un nombre premier est un entier >2 divisible uniquement par 1 et lui même. Deux nombres premiers sont dits jumeaux s'ils ne diffèrent que de 2.

Exemple :
M=3, N=20
La fonction affichera :

(3, 5)
(5, 7)
(11, 13)
(17, 19)

Rappel sur l'algorithmique et le langage C

Programme équivalent en C:

```
void main()
{
type déclaration_de_variable;

actions_à_réaliser;

}
```

La structure de sélection simple

```
if(conditions)
{
actions_à_réaliser;
}
else
{
autres_actions_à_réaliser;
}
```

Structure de répétition (nombre de répétition connu)

```
//on suppose que l'on a écrit plus haut:
int var;

for(var = valeur1;
var<=valeur2;var=var+1)
```

Structures de répétition (nombre de répétition inconnu)

```
while(conditions)
{
actions_à_répéter;
        modification_dela_condition;
}
```

Déclaration de fonction

```
type_Fonction Nom_Fonction(type1
argument1,
type2 argument2, ...)
{type variable_sortie ;
liste d'instructions;
return variable_sortie ;
/*return permet de renvoyer la valeur de
```

Déclaration de fonction qui ne renvoie rien

```
void Nom_Fonction(type1 argument1,
type2 argument2, ...)
{
liste d'instructions;
/*ici on n'a pas besoin d'utiliser la
fonction return car la fonction n'a pas
```

Ecriture / Lecture

printf(" voici un entier: %d\n",a); /* si on veut afficher la variable a qui est un int (entier), sinon on met %f pour un float (réel)*/
Equivalent en langage algorithme: Ecrire('voici un entier',a)
scanf("%d,%f",&a,&x); /* si l'on veut donner des valeurs aux variables a et x: a est un entier et x est un réel*/
Equivalent en langage algorithme: Lire(a,x)

**Déclaration de tableau*

En langage C, la déclaration d'un tableau (ici un tableau de 20 entiers par exemple) se fait comme suit:

```
int Tab[20]; //attention en langage C, les éléments du tableau vont de Tab[0] à Tab[19]
```