Algorithmique et programmation
M2 CCI
Contrôle continu

23 Novembre 2007

Indications générales

Les calculatrices et les téléphones portables sont *strictement interdits.*
Les fonctions qui testent une condition doivent renvoyer un entier valant 0 si la condition est fausse, 1 sinon.
Il est bien entendu possible d'écrire des fonctions intermédiaires pour répondre à la question posée.
La lisibilité du code sera prise en compte dans la notation.
Un code commenté est recommandé.

Chaque fonction demandé devra être testé dans le main. Ainsi, **aucune fonction demandé dans ce sujet ne doit comporté de scanf ou de printf. C'est dans le main que se fera les entrées-sorties.**

Barème indicatif :
- Exercice 1 : 4 pts ;
- Exercice 2 : 6 pts ;
- Exercice 3 : 7 pts ;
- Exercice 4 : 3 pts ;

- **Les points seront divisés entre la "bonne" compilation de la fonction, la "justesse" de la fonction, l'esthétique de la fonction et son utilisation dans le main.**

1 Exercices

1.1 Le gain tordu

Un homme gagne un prix de 50 euros sur un jeu télévisé. Le présentateur lui explique :
Vous avez gagné 50 euros. ***Mais****, si vous le voulez, vous pouvez doublez vos gains ! Pour cela, il suffit de les remettre en jeu et participer à un mini-jeu. Dans ce mini-jeu, vous avez 3 minutes et si vous arrivez à courir à cloche pied 30 métres vous conservez vos 50 euros. Pour chaque 10 métres supplémentaires, on vous ajoute 10 euros. Au maximum, vous pourrez avoir le double, soit 100 euros ! Par contre, si vous n'arrivez pas à faire au minimum 30 mètres, vous perdez tout !*

- Écrire une fonction **courir** qui calcule, en fonction du nombre de mètres que court l'homme, son prix total.
- Écrire un programme qui demande à l'utilisateur combien de mètres l'homme a-t-il couru et affiche le prix total.

Exemples :

```
Combien ?
5
Le prix total est 0 euros.

Combien ?
33
Le prix total est 50 euros.

Combien ?
52
Le prix total est 70 euros.

Combien ?
1234
Le prix total est 100 euros.
```

1.2 Division par 3

On dit qu'un nombre est divisible par 3 si la somme de ces chiffres donne un chiffre divisible par 3.

Remarquons que par exemple 39 est divisible par 3 car la somme de ses chiffres vaut 12. Et 12 est divisible par 3 car la somme de ces chiffres est égale à 3. On sait que 3 est divisible par 3 donc 21 est divisible par 3 et donc 39 est divisible par 3.

- Ecrire une fonction qui retourne la somme des chiffres d'un nombre ;
- Ecrire une fonction qui calcule la somme des chiffres d'un nombre et, si celle-ci est au-dessus de 10, refera le calcul jusqu'à obtenir un nombre en dessous de 10. Cette fonction retournera donc un nombre entre 0 et 9. Par exemple, si on donne 39 à cette fonction, elle retournera 3 ;
- Ecrire un programme qui demande à l'utilisateur un nombre et lui dit s'il est divisible par 3.

Exemple :

```
Le nombre ?
995
Le nombre 995 n'est pas divisible par 3

Le nombre ?
996
Le nombre 996 est divisible par 3

Le nombre ?
123456
Le nombre 123456 est divisible par 3
```

1.3 Approximation d'une racine

Soit la suite A_n qui calcule une approximation de la racine de X et définie comme ceci :

$A_n = (A_{n-1} + X/A_{n-1})/2.0$

$A_1 = X$

- Ecrire une fonction *approx_iter* qui calcule le n-ième terme de la suite utilisant une boucle ;
- Ecrire une fonction *approx_iter* qui calcule le n-ième terme de la suite utilisant une fonction récursive ;
- Ecrire un programme qui demande le nombre X et le nombre n et affiche le n-ème terme de la suite A_n racine.

Exemple :

```
Le nombre x et le nombre n ?
5000 1
L'approximation de 5000.000000 au rang 1 est :
        Iteratif : 5000.000000,
        Recursif : 5000.000000

Le nombre x et le nombre n ?
5000 350
L'approximation de 5000.000000 au rang 350 est :
        Iteratif : 70.710678,
        Recursif : 70.710678
```

1.4 Nombre parfait

Le programme suivant devrait demander un nombre n à l'utilisateur et afficher les nombres parfaits de 1 à n. Un nombre est dit parfait s'il est égal à la somme de ces diviseurs. Voici le programme :

http ://icps.u-strasbg.fr/ beyler/TPTD/20072008/Algo/TPnote/nbrparfait.c

Lorsqu'il est compilé, il devrait fonctionner ainsi :

```
Donner un nombre :
500
Les nombres parfaits sont :
1
6
28
496
```

- Corriger le programme pour qu'il fonctionne comme ci-dessus.

Remarque : En corrigeant chaque erreur de compilation utilisant les options -Wall -Wextra -O2, le programme devrait fonctionner.