TP 5 – fonctions

1. **Ration calorique.** La ration calorique exprime le nombre de calories dont a besoin un individu pendant 24 heures. Les lignes qui suivent indiquent une façon simplifiée de la calculer. La ration calorique est la somme du métabolisme de base et du métabolisme d'activités.

- Le métabolisme de base Mb est le nombre de calories nécessaires à l'organisme pendant 24 heures, le sujet restant sans activité (par exemple, si le sujet reste couché). Sa valeur peut être obtenue à l'aide de la formule simplifiée : Mb = 24 * α * SC, où
  - SC est la surface corporelle : SC = 0.2 * $P^{0.4}$ * $T^{0.7}$, P désignant la masse (en kg), T la taille (en mètre) ;
  - et α est un coefficient dépendant de l'âge, valant : 45 pour un sujet dont l'âge est compris entre 10 et 15 ans, 40 entre 15 et 20 ans, 35 entre 20 et 60 ans, 30 au-delà de 60 ans.

Par ailleurs, une femme, à taille, masse et âge égaux, a un métabolisme de base inférieur de 10% à celui d'un homme. Mais une femme enceinte ou qui allaite a besoin d'absorber un supplément d'environ 600 calories.

L'archive disponible sur la plateforme contient un fichier ex1.c dans le quel vous trouverez un squelette permettant de valider les premières fonctions. *–correction disponible–*

Vous coderez les fonctions suivantes dont voici les prototypes et leurs commentaires au format Doxygen.

```
/**  Calcule la surface corporelle
  @param t : taille en m
  @param m : masse en kg
  @return la surface corporelle */
float surfaceCorporelle (float m, float t) ;

/**  Calcule le metabolisme de base
  @param sc : surface corporelle
  @param a : age
  @return le metabolisme de base */
float metaBase (float sc, int a) ;

/**  prise en compte du sexe dans le metabolisme de base
  @param mb : le métabolisme initial
  @param sexe : 0 si homme, 1 si femme
  @param allaitmt : 1 si oui, 0 sinon
  @return le metabolisme de base modifié */
??? metaBaseSexe ( ????? )
```

Vous aurez besoin de la fonction powf. Consultez le manuel en ligne en tapant *man powf* dans la console.
Vous aurez également besoin de compiler en précisant au compilateur que vous utilisez la librairie mathématique standard de cette manière :
gcc prog.c -o prog.bin -lm

Travail demandé :
A partir du squelette de l'archive, compléter les codes des trois fonctions afin de valider les tests unitaires suivants :

```
sc = surfaceCorporelle (55, 1.7) ; // doit retourner 1.440421
printf(" sc =%f \n", sc ) ;
mb = metaBase (sc, 20) ; // doit retourner 1209.953
printf(" mb =%.3f \n", mb ) ;
mbs = metaBaseSexe (mb , 1, 1 )  ;  // doit retourner 1688.958
printf(" mbs =%.3f \n", mbs ) ;
```

2. **Soit** le programme principal suivant (disponible dans l'archive) :

```
/**
 * @file ex2.c
 * @author : Philippe Lefebvre <philippe.lefebvre@ensicaen.fr>
 * @date : 2020/12/25
 * @brief : Programme de test des fonctions trier3entier et afficher3entiers
 * @version : 1.0
**/

#include<stdio.h>
int main (void) {
    int a=7, b=4,c=5 ;
    // trier3Entiers (&a, &b, &c) ;
    afficher3entiers(a, b, c) ;
}
```

2.1. **Écrire** la fonction `void afficher3entiers(int va, int vb, int vc)` qui affiche à l'écran les valeurs des 3 nombres va, vb, vc. Testez votre fonction.

2.2. **Écrire** la fonction `void trier3Entiers(int *pa, int *pb, int *pc)` qui trie 3 nombres stockés dans trois variables dont les adresses *pa, pb* et *pc* sont passées en paramètre. A la fin de la fonction, la variable pointée par *pa* contiendra le plus petit et la variable pointée par pc contiendra le plus grand. Tester vos 2 fonctions avec le `main( )` fourni.

2.3. **Écrire** la fonction `demander3Entiers( )` qui demande 3 entiers à l'utilisateur et qui retourne ces 3 nombres dans trois variables dont les adresses sont passées en paramètre. A vous de trouver le prototype adéquat.

2.4. **Écrire** la fonction `main( )` permettant de tester la fonction `demander3Entiers( )`. *–correction disponible–*

3. **Reprendre** l'exercice 1 et compléter le programme pour déterminer les besoins énergétiques d'un individu corrigé par le métabolisme d'activités. Ce dernier dépend de l'activité physique du sujet au cours des 24 heures d'une journée type. On distinguera :
   - le sommeil ou repos couché qui requiert 0 calorie ;
   - l'activité sédentaire (position assise) qui requiert 50 calories par heure ;
   - l'exercice léger (la marche par ex.) qui requiert 100 calories par heure ;
   - le sport qui requiert 200 calories par heure.

   Écrire le programme qui comportera un dialogue homme-machine sous formes de questions – réponses portant sur les caractéristiques de type masse, âge, activité physique quotidienne, etc. La fonction main( ) ne comportera que l'appel à la fonction ihm( ).

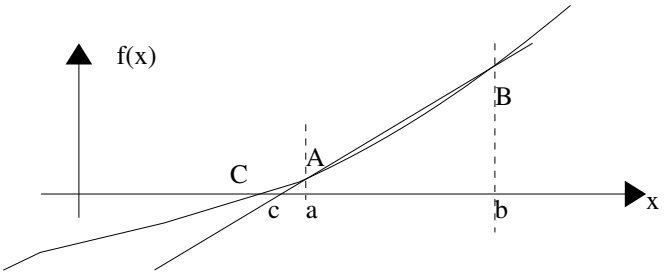
4. **Programmez** le calcul de la suite de Fibonacci de manière récursive.

5. **Recherche par dichotomie. Implémentez de manière récursive** l'algorithme de recherche par dichotomie du zéro de la fonction
   $x \rightarrow x^3 + 3.x^2 – 598.x – 1200$
   sur l'intervalle [0, 100] , à une précision de 10^{-9}.
   On programmera f(x) de cette manière : *f(x) = -1200.0 +x*(-598.0 +x*(3+ x))*
   – Pourquoi est-ce plus efficace ?

   Rappel : soit f une fonction continue sur un intervalle [a,b] et s'annulant exactement une fois sur cet intervalle. Pour trouver ce zéro, l'algorithme de recherche par dichotomie consiste à :
   – découper l'intervalle courant [g,d] en deux intervalles égaux [g,m] et [m,d] (avec m milieu de g et de d) ;
   – à retenir celui de ces deux intervalles sur lequel f s'annule (i.e. les valeurs de f aux extrémités de cet intervalle sont de signes opposés) ;
   – jusqu'à obtention de la précision voulue par l'écart d – g des extrémités de l'intervalle courant.
   –correction disponible–

6. **Méthode de Newton Raphson**. C'est une autre méthode de recherche du zéro d'une fonction. La méthode est la suivante : Soit un point A, proche du zéro à trouver. On calcul pour cela un point C, intersection de la tangente en A avec l'axe des abscisses. Puis on réitère le processus en C.
   Une méthode approchée, appelée méthode des sécantes, consiste à utiliser un point B, proche de A et de déterminer C comme étant l'intersection de (AB) avec l'axe des abscisses.

$$c = b + f(b) \cdot \frac{a-b}{f(b)-f(a)}$$



On réitère le processus en renommant C en A et A en B, puis on détermine un nouveau point C.

Programmez cette méthode sur le polynôme de l'exercice précédent.

Cet algorithme est-il plus efficace que la recherche par dichotomie ?