Exercice B0010 *Calculs avec les flottants*

Cet énoncé est accompagné d'un ou plusieurs codes compagnons en C fournissant certaines des fonctions mentionnées dans l'énoncé : il sont à compléter en y implémentant les fonctions demandées.

La ligne de compilation `gcc -o main.exe -Wall *.c -lm` *vous permet de créer un exécutable* `main.exe` *à partir du ou des fichiers C fournis. Vous pouvez également utiliser l'utilitaire* `make`*. En ligne de commande, il suffit d'écrire* `make`*. Dans les deux cas, si la compilation réussit, le programme peut être exécuté avec la commande* `./main.exe`*.*

Il est possible d'activer davantage d'avertissements et un outil d'analyse de la gestion de la mémoire avec la ligne de compilation `gcc -o main.exe -g -Wall -Wextra -fsanitize=address *.c -lm` *ou en écrivant* `make safe`*. L'examinateur pourra vous demander de compiler avec ces options.*

Si vous désirez forcer la compilation de tous les fichiers, vous pouvez au préalable nettoyer le répertoire en faisant `make clean` *et relancer une compilation. Le fichier* flottants.c *fournit les structures de données et certaines fonctions mentionnées dans l'énoncé.*

Un nombre réel x est représenté en machine en base 2 par un flottant qui a un signe s, une mantisse m et un exposant e tel que $x = s \times m \times 2^e$. Dans la norme IEEE 754, en convention normalisé la partie entière de la mantisse est 1 qui est un bit caché. En simple précision, le signe est codé sur 1 bit, la partie décimale de la mantisse sur 23 bits et l'exposant sur 8 bits. En double précision, le signe est codé sur 1 bit, la partie décimale de la mantisse sur 52 bits et l'exposant sur 11 bits.

Dans cet exercice, on observe le résultat de calculs obtenus par un programme. On pourra utiliser la fonction de signature `double pow(double v, double p)` qui calcule v^p.

1. Dans la fonction principale `main`, on a défini 3 variables a, b, c de type `double`. Compléter le code pour calculer et afficher le résultat des opérations $(a+b)+c$ et $a+(b+c)$. Que constatez-vous?
2. Compte tenu des approximations faites lors du codage, on peut trouver plusieurs nombres x tels que $1+x=1$ après un calcul fait par la machine. Le plus petit nombre représentable exactement en machine et supérieur à 1 s'écrit $1+\epsilon$, avec ϵ un réel appelé ϵ machine. On admet que ϵ s'écrit sous la forme 2^{-n} avec n un entier naturel strictement positif. Écrire une fonction de signature `int epsilon()` qui renvoie la valeur de n. Justifier cette valeur.
3. On considère une suite $(u_n)_{n\in\mathbb{N}}$ définie par

$$\begin{cases} u_0 &= 2 \\ u_1 &= -4 \\ u_n &= 111 - \frac{1130}{u_{n-1}} + \frac{3000}{u_{n-1}\times u_{n-2}} \text{ si } n \geq 2 \end{cases}$$

   Écrire une fonction de signature `double u(int n)` qui renvoie la valeur du terme u_n.
4. La limite théorique de la suite $(u_n)_{n\in\mathbb{N}}$ est 6. Compléter la fonction `main` afin d'afficher les 22 premiers termes de la suite. Vers quelle valeur semble tendre la suite?
5. On définit une liste chaînée de nombres à l'aide d'une structure `nb` comportant un double et un pointeur vers une structure `nb` définie ci-dessous

```
struct nb {double x; struct nb* suivant;};
```

   Écrire une fonction de signature `double somme(struct nb* tab)` qui calcule la somme des éléments de la liste `tab`.
6. L'algorithme suivant permet d'augmenter la précision du calcul lors du calcul d'une somme.

Entrée : Une liste l de réels triée dans l'ordre croissant de taille au moins 2.
Sortie : La somme des réels contenus dans la liste l.
tant que la liste l contient strictement plus d'un élément
  Calculer la somme $s = x + y$ des deux premiers éléments x et y de l
  Supprimer x et y de l
  Insérer s dans l de sorte à ce que l reste triée
renvoyer l'unique élément de l

- Compléter la fonction `somme2` qui implémente cet algorithme. *Remarque de JPeig : Le code fourni comporte plusieurs incohérences et ne permet pas de bien structurer le programme... On pourra corriger et compléter cette fonction, ou s'en inspirer pour en écrire une autre en partant de zéro.*
- La fonction proposée ne prend pas en compte un cas d'insertion. Illustrer ce propos.