Contrôle Machine
DEV5.1 Qualité algorithmique

Consignes

- Créez un repo git ayant pour nom CONTROLE_DEV51_<login>
- Tous les codes doivent être écrit en C
- Le contrôle est individuel
- Droit d'utiliser ses cours et autres TD
- Mettez un fichier réponse sous format TXT / PDF / MD dans le git
- Mettez le code source uniquement (.c et .h autorisé) dans le git
- Mettez un fichier readme donnant la commande de compilation du programme dans le git

Ex 1 - Racine carrée

- Créez une fonction "racineCarree()" qui renvoie la racine carrée d'un nombre entier positif si la racine est aussi un nombre entier, ou -1 si le nombre n'a pas de racine entière

  racineCarree(9) → 3
  racineCarree(10) → -1

  Aide : La racine d'un nombre positif X est compris entre 0 et X inclus

- Créez ensuite une fonction "racineCarreeTab()" qui renvoie la racine carrée d'un ensemble de nombre entier positif. Elle renverra aussi -1 si le nombre n'a pas de racine entière. Cette fonction peut avoir plusieurs arguments si nécessaire, la taille du tableau peut être variable.

  racineCarreeTab([9,25,4]) → [3,5,2]
  racineCarreeTab([10,36,2]) → [-1,6,-1]

Ex 2 - Qualité de code (1)

- Profiler l'appel à racineCarreeTab, vous pouvez prendre comme donnée de test un tableau d'une taille de 10.000 grand entiers(disons > 1.000.000)

- Calculer la complexité cyclomatique de racineCarree() et racineCarreeTab()

- Calculer la complexité algorithmique de racineCarree() et racineCarreeTab()

Ex 3 - Tri Spécial

- Créez une fonction “TriSpecial()” qui trie un tableau selon une règle un peu particulière :
    - Si la quantité de racine carrée non entière du tableau est paire, alors le tableau sera trié dans l’ordre suivant :

```
[
    Nombre(0),
    Somme(tableau)*Nombre(1),
    Nombre(2),
    Somme(tableau)*Nombre(3),
    ...
]
```

    - Sinon, le tableau sera trié ainsi :

```
[
    racineCarree(Nombre(0)),
    Somme(racineCarree(tableau))*Nombre(1),
    racineCarree(Nombre(2)),
    Somme(racineCarree(tableau))*Nombre(3),
    ...
]
```

    Réutilisez la fonction racineCarree() développé à l’Ex1

    TriSpecial([3,5,25,16]) → [3,245,25,784] // Deux racines non entières
    TriSpecial([36,9,100,2,3,7]) → [6,144,10,32,-1,112] // Trois racines non entières

Ex 4 - Qualité de code (2)

- Profiler l’appel à TriSpecial, vous pouvez prendre comme donnée de test un tableau d’une taille de 10.000 grand entiers(disons > 1.000.000)

- Calculer la complexité cyclomatique de TriSpecial

- Calculer la complexité algorithmique de TriSpecial