L3 Nantes Université
X31I050 2022/2023

Feuille de travaux pratiques n° 1
Représentation de l'information

On réalisera le travail demandé en C ou C++ au choix.

Exercice 1.1

Écrire une fonction `base_converter()` prenant en entrée une chaîne de caractères (`string` C++ ou `char[]` C), une base de départ, une base d'arrivée, et retournant une chaîne de caractères correspondant à la chaîne initiale exprimée dans la base d'arrivée. On considérera que les bases de départ et d'arrivée sont comprises entre 2 et 36 inclus.

Exemple d'utilisation :

```
base_converter("3eh12",18,30) -> "eqne"
```

On prendra soin de gérer les cas d'erreurs.

Exercice 1.2

1. Écrire une fonction `bin2brgc()` transformant un entier binaire non-signé en code de Gray BRGC ;
2. Écrire une fonction `brgc2bin()` transformant un code de Gray BRGC en entier binaire non-signé.

Les deux fonctions prendront en entrée un entier `e` et une taille `t`. On affichera le résultat sous forme décimale et binaire.

Exemple d'utilisation :

```
bin2brgc(4,3) -> 6 (110)
brgc2bin(7,3) -> 5 (101)
```

Exercice 1.3

On souhaite calculer le plus précisément possible une somme S de nombres flottants en double précision se trouvant dans un tableau `T` de taille n :

$$S = \sum_{i=0}^{n-1} T_i$$

Pour cela, on va comparer expérimentalement la qualité de différents algorithmes de sommation.

1. **Somme récursive.** Il s'agit de l'algorithme le plus simple, où l'on ajoute chacun des T_i dans un accumulateur dans l'ordre où ils apparaissent dans le tableau. Écrire la fonction `somme_recursive()` prenant en paramètre un tableau `T` et retournant la somme de ses éléments (Attention : la fonction est purement itérative et **non récursive**) ;
2. **Somme récursive inverse.** On inverse le tableau `T` et l'on applique une somme récursive sur le nouveau tableau. Écrire la fonction `somme_recursive_inverse()` prenant en paramètre un tableau `T` et retournant la somme de ses éléments dans l'ordre inverse de leurs positions ;
3. **Somme en valeurs (dé)croissantes.** On trie le tableau `T` dans l'ordre (dé)croissant des valeurs, puis l'on applique l'algorithme de somme récursive.
   (a) Écrire la fonction `somme_croissante()` évaluant la somme des éléments du tableau `T` passé en paramètres pour un tri en ordre croissant ;

(b) Écrire la fonction `somme_decroissante()` évaluant la somme des éléments du tableau `T` passé en paramètres pour un tri en ordre décroissant.

4. **Somme en valeurs absolues (dé)croissantes.** On trie le tableau `T` dans l'ordre (dé)croissant des valeurs absolues, puis l'on applique l'algorithme de somme récursive.

   (a) Écrire la fonction `somme_abs_croissante()` évaluant la somme des éléments du tableau `T` passé en paramètres pour un tri en ordre croissant des valeurs absolues ;

   (b) Écrire la fonction `somme_abs_decroissante()` évaluant la somme des éléments du tableau `T` passé en paramètres pour un tri en ordre décroissant des valeurs absolues.

Tester les fonctions écrites avec le fichiers de données `donnees_somme.h` ou `donnees_somme.hpp` se trouvant sur *madoc*. Justifier les observations faites. Déterminer les points forts et les points faibles de chaque méthode ; vérifier les conclusions à l'aide de jeux de données à définir. Proposer d'autres méthodes de sommation précises.