TD 5

Sur les machines classiques, le type int peut représenter des entiers sur 32 bits soit de -2^{31} à $2^{31}-1$ ($2^{31} \equiv 2$ milliards). Pour utiliser des nombres entiers plus grands, l'utilisateur doit définir par logiciel le traitement sur ces "grands nombres" appelés les **bignums**. Ce TD propose d'implanter la structure pour le type bignum et les opérations de bases (addition, soustraction, multiplication) travaillant sur ce type.

On utilisera les types suivants :

```
/* Taille maximum d'un bignum */
#define TAILLE_MAX 10

/* Signe + / - */
typedef enum {PLUS = 0,MOINS = 1} signe;

typedef unsigned short int chiffre;

typedef struct bignum {
  int longueur;                 /* Le nombre de chiffres significatifs */
  int taille;                   /* nombre maximal de chiffre pour ce bignum */
  signe signe;                  /* Le signe du bignum */
  chiffre num[TAILLE_MAX];      /* La représentation du bignum */
} bignum_t;
```

Les bignums seront représentés par la structure `bignum`. Le champ `longueur` indique le nombre de chiffres significatifs. Le champs `taille` indique l'espace mémoire réservé pour le nombre. Le champ `signe` indique le signe (`PLUS/MOINS`) du nombre. Enfin le champ `num` contient tous les chiffres du bignum tel que num[i] soit le chiffre correspondant à son rang (unité en 0, dizaine en 1, ...).

1. Ecrivez une fonction appelée `initialiser` qui renvoie une structure avec l'ensemble des valeurs de la structure `bignum` initialisées. On initialisera la longueur à -1.

2. Ecrivez une fonction appelée `lire` qui renvoie la structure bignum lue au clavier. Cette fonction devra lire au clavier un signe ('+' ou '-' obligatoirement présent) suivit d'une suite de chiffres compris entre '0' et '9' pour les mettre dans la structure bignum. On utilisera un tableau temporaire pour réaliser l'inversion des chiffres après les avoir lu du clavier. On utilisera `getchar()` pour lire les caractères du clavier.

   ```
   Exemple : Clavier : -9456 donne
   struct bignum {
     longueur = 4;
     signe = MOINS;
     num = {6, 5, 4, 9};
   }
   ```

3. Ecrire une fonction appelée `ecrire` qui affiche la valeur d'un bignum sur l'écran.

4. Ecrire une fonction `neg` qui prend en argument un bignum et qui retourne son opposée.
5. Ecrire la fonction `addition_aux` pour *des bignums de même signe*. Cette fonction doit vérifier que les deux bignums ont le même signe pour faire le calcul et doit renvoyer une erreur sinon.
6. Ecrire la fonction `soustraction_aux` pour *des bignums de même signe*.
7. Améliorer les fonctions `addition` et `soustraction` pour traiter le cas des 2 arguments de signes différents.
8. Ecrire une fonction `puissance10` qui prend un bignum et un entier en argument. Cette fonction doit calculer bignum*10^{entier}.
9. Ecrire la fonction `multiplication`. Vous utiliserez la fonction `addition` et/ou la fonction `puissance10`.
10. (facultatif) Ecrire la fonction `division` entière qui divise un bignum par un autre.