Travaux Dirigés de programmation n°4

Cours de Programmation C

—L2.1—

▸ **Exercice 1.** *Opérateurs binaires*

1. *Donner la représentation en bits des deux variables suivantes :*

   ```
   char a = 7;
   char b = 25;
   ```

2. *Ecrire les représentations de* `a&b`, `a|b`, `a^b` *et les valeurs correspondantes.*
3. *Effectuer lignes à lignes les opérations de la fonction suivante avec les variables* `a` *et* `b` *précédentes et conclure sur ce que fait cette fonction.*

   ```
   void S(char *a, char *b) {

       (*a) = (*a)^(*b);
       (*b) = (*a)^(*b);
       (*a) = (*a)^(*b);

   }
   ```

▸ **Exercice 2.** *Lecture et écriture bit à bit*

On veut à présent lire et écrire des bits á l'intérieur d'un octet. En language C, les octets sont représentés par des variables `char`*, il n'y a pas de type spécifique pour les bits.*

1. *Écrire la fonction* `char recupereBitChar(char octet, char p)` *qui retourne le bit en position* `p` *dans* `octet`*. La valeur retournée est un* `char` *égal à 1 ou 0. La position 0 correspond au bit de poids fort et 7 au bit de poids faible.*
2. *Écrire la fonction* `void placeBitChar1(char *octet, char p)` *qui force un bit 1 dans* `*octet` *à la position* `p`*.*

   Exemple : Si `*octet= 10110000` *et* `p = 6`*, alors* `*octet` *doit être modifié en* `10110100`*.*
3. *Écrire la fonction* `void placeBitChar0(char *octet, char p)` *qui force un bit 0 dans* `*octet` *à la position* `p`*.*
4. *On souhaite compresser une liste d'entiers dont on sait les valeurs proches les unes des autres : plutôt que de coder chaque entier séparément, on codera le premier entier puis les différences successives. Par exemple :* `789 790 788 789 789 792` *devient* `789 +1 -2 +1 0 +3`*. Cette seconde liste sera codée par une liste de bits :*
   – *4 octets correspondants au premier entier*

– *1 octet correspondant au nombre* `n` *de bits nécessaires pour coder la valeur absolue maximale des différences.*
– *la liste des différences sur* `n+1` *bits chacune (premier bit pour le signe puis* `n` *bits pour la valeur absolue).*

Ainsi la liste précédente devient :

```
00000000 00000000 00000011 00010101 00000010 0 01 1 10 0 01 0 00 0 11
```

Écrire la fonction `size_t compresse(int * valeurs, char * comp, int nbValeurs)` *qui compresse la liste* `valeur` *dans* `comp` *(supposé de taille suffisante) et retourne le nombre d'octets utlisiés.*

5. *Écrire la fonction* `void decompresse(int * valeurs, char * comp, int nbValeurs)`