TD opérations binaires

Rappel:

Opérations binaires en C:

Le et (and) à pour symbole : &
Le ou (or) à pour symbole : |
L'inverse (not) à pour symbole : ~
Le ou exclusif (xor) à pour symbole : ^
Décalage à droite : >>
Décalage à gauche : <<

Tables de vérité:

AND (&)	**OR (|)**	**NOT (~)**	**XOR(^)**
1 & 1 = 1	1 | 1 = 1	~0 = 1	0 ^ 0 = 0
0 & 0 = 0	0 | 0 = 0	~1 = 0	1 ^ 1 = 0
0 & 1 = 0	0 | 1 = 1		0 ^ 1 = 1
1 & 0 = 0	1 | 0 = 1		1 ^ 0 = 1

Exemple sur 8 bits:

Soit deux variables codées sur un octet: a = 7 et b = 28

a = 00000111 **et** b = 00011100

. a & b →
```
00000111
00011100
--------
00000100
```

. a | b →
```
00000111
00011100
--------
00011111
```

. ~a = 11111000 et . ~b = 11100011

. a ^ b →
```
00000111
00011100
--------
00011011
```

. a >> 2 = 00000001

. a << 5 = 11100000

. b >> 3 = 00000011

. b << 1 = 00111000

Exemple de fontions en C:

```
unsigned char AND(unsigned char a, unsigned char b) { return (a & b); }
unsigned char OR(unsigned char a, unsigned char b) { return (a | b); }
unsigned char XOR(unsigned char a, unsigned char b) { return (a ^ b); }
unsigned char SHL(unsigned char a, unsigned char b) { return (a << b); }
unsigned char SHR(unsigned char a, unsigned char b) { return (a >> b); }
```

Exo 0x00:

. Écrire une fonction en C qui prend en paramètre un entier non signé codé sur 8 bits (1 octet) et qui inverse l'ordre de ses bits.

Exemple:

7 = 00000111 en binaire

inv(7) devra retourner 224

224 = 11100000 en binaire

Exo 0x01:

. Tracer les différentes étapes (en binaire ou en hexadécimal) de la fonction suivante pour v = 1, v = 7 et v = 15:

```
unsigned char pop(unsigned char v)
{
        v = (v & 0x55) + ((v >>  1) & 0x55);
        v = (v & 0x33) + ((v >>  2) & 0x33);
        v = (v & 0x0F) + ((v >>  4) & 0x0F);

        return v;
}
```

. Que fait cette fonction?

Exo 0x02:

. Écrire une fonction en C qui prend en paramètre deux entiers **a** et **r** codés sur 8 bits et qui effectue une rotation à gauche de **r** positions des bits de **a**.

Exemple:

8 = 00001000 en binaire

rot_left(8, 5) = 1

1 = 00000001 en binaire

. Écrire la fonction qui effectue une rotation à droite.

rot_right(8, 5) = 64

64 = 01000000 en binaire

Exo 0x03:

. Écrire une fonction en C qui prend en paramètre un nombre codé sur 8 bits et qui aggrège tous les bits à 1 à droite et tous les bits à 0 à gauche.

Exemple:

0x44 en hexadécimal = 01000100 en binaire
0xF8 en hexadécimal = 11111000 en binaire

f(0x44) = 0x03 = 00000011 en binaire
f(0xF8) = 0x1F = 00011111 en binaire

Exo 0x04:

. Quel est le résultat des opérations suivantes sachant que a et b sont des entiers différents, non signés et codés sur 8 bits?

```
a =  a ^ b;
b = a ^ b;
a = a ^ b;
```

. Tester pour a = 3 et b = 9.

Que fait cet algorithme?

Exo 0x05:

Nous disposons de la séquence génétique suivante: **S0** = ATGTTTGCCACCTATC.
Chaque caractère de la séquence est associé à un encodage binaire:

A = 00, T = 11, G = 01, C = 10

1) Donner la séquence binaire de la séquence génétique.
2) Calculer la distance de Hamming entre la séquence S0 et la séquence suivante:
S1 = ATGTTGGCCACGTATC
3) Y a-t-il des erreurs?
4) Sachant que A et le réciproque de T et que G est le réciproque de C, donner la séquence binaire de la séquence réciproque de **S0**.
5) Effectuer un XOR entre la séquence binaire de **S0** et celle de sa réciproque, puis inverser (~) les bits du résultat obtenu.
6) Que constatez-vous?