Exercices

1. **Expression booléenne :**
   a, b et c sont trois valeurs booléennes.
   Dans quels cas les expressions proposées sont-elles vraies ?
   On dressera une table de vérité pour étayer la réponse :
   (a) a et (non b)
   (b) (non a) et (non b)
   (c) a ou (b et c)

2. **XOR - OU Exclusif** :
   Sachant que : a XOR b = (a et (non b)) ou (b et (non a))
   Écrire la table de vérité de la fonction XOR.

3. Toutes les égalités suivantes sont vraies pour tous booléens a, b et c.
   — **Complémentarité** :
   a ou (non a) = 1 a et (non a) = 0 non (non a) = a

   — **Commutativité** :
   a ou b = b ou a a et b = b et a

   — **Associativité** :
   (a ou b) ou c = a ou (b ou c) = a ou b ou c
   (a et b) et c = a et (b et c) = a et b et c

   — **Distributivité** :
   a et (b ou c) = (a et b) ou (a et c)
   a ou (b et c) = (a ou b) et (a ou c)

   (a) Démontrer les deux égalités de la distributivité à l'aide des tables de vérités suivantes :

a	b	c	b ou c	a et (b ou c)
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

a	b	c	a et b	a et c	(a et b) ou (a et c)
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

a	b	c	b et c	a ou (b ou c)
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

a	b	c	a ou b	a ou c	(a ou b) et (a ou c)
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

(b) Le mathématicien britannique Auguste de Morgan (1806–1871) a également contribué à développer l'algèbre de Boole à l'aide de son théorème :
non (a ou b) = (non a) et (non b)
non (a et b) = (non a) ou (non b)

Démontrer ce théorème à l'aide des tables ci-dessous :

a	b	a ou b	non (a ou b)
0	0		
0	1		
1	0		
1	1		

a	b	non a	non b	(non a) et (non b)
0	0			
0	1			
1	0			
1	1			

a	b	a et b	non (a et b)
0	0		
0	1		
1	0		
1	1		

a	b	non a	non b	(non a) ou (non b)
0	0			
0	1			
1	0			
1	1			

4. **Une application des opérations booléennes : les masques de sous-réseau :**
Dans un réseau TCP/IP, un ordinateur a une adresse IP qui l'identifie de manière unique (comme un numéro de téléphone), ainsi qu'un masque de sous-réseau.
L'adresse IP et le masque de sous-réseau sont des groupes de 4 entiers positifs, codés tous les deux sur 4 octets.
Le masque identifie à quel sous réseau d'Internet il fait partie, et permet d'obtenir l'adresse IP du sous-réseau à l'aide d'une opération booléenne.

Exemple : Un ordinateur a pour IP 192.168.0.42, et son masque de sous-réseau est 255.255.255.0.
Pour obtenir l'adresse du sous-réseau de l'ordinateur, on effectue alors un AND entre les octets de l'IP et les octets du masque.
Sur un processeur 32 bits, le AND est une opération primitive du processeur, donc l'opération de détermination du sous-masque réseau se fait en une instruction.

(a) Écrire en binaire les 4 octets de l'adresse IP et du masque de sous-réseau.

(b) En déduire l'expression binaire de l'adresse du sous-réseau.

(c) Convertir cette adresse sous sa forme décimale.

5. **TP : simulation de circuit logique : additionneur binaire**

   (a) Se rendre sur le site `https://profjahier.github.io/simcirjs/cimcirjs.html` ou sur une version épurée fournie par votre prof.

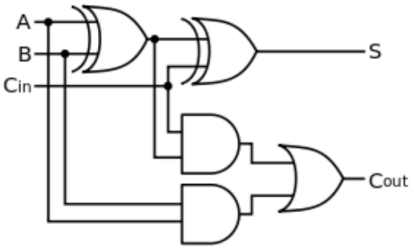
   (b) Un demi additionneur permet d'ajouter deux bits A et B. La sortie principale S vaut 1 si A ou B seulement vaut 1, et la retenue C vaut 1 si A et B valent 1.

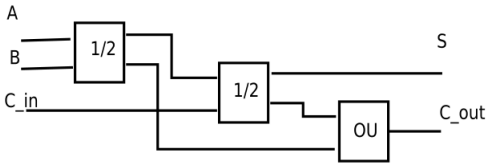
   i. Quelle fonction booléenne correspond à la sortie S ?

   ii. Quelle fonction booléenne correspond à la retenue C ?

   iii. Câbler un demi additionneur puis vérifier son fonctionnement.
   Les entrées A et B sont matérialisées par des interrupteurs (connectés à l'alimentation DC). Les sorties S et C sont matérialisées par des DEL.

   (c) Un additionneur complet prend une retenue entrante en plus des deux entrées A et B.
   Son logigramme est donné ci-dessous :

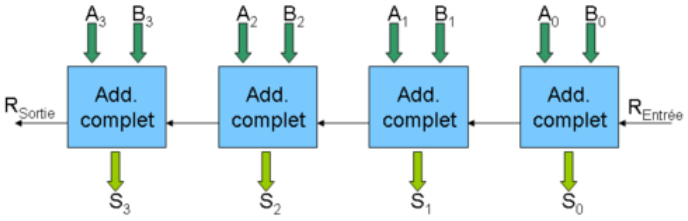




   i. Établir sa table de vérité.

   ii. Câbler un additionneur puis vérifier son fonctionnement.
   On pourra (non obligatoire) utiliser le module halfadder fourni dans la bibliothèque.

   (d) Un additionneur 4 bits permet l'addition de 2 mots de 4 bits ($A_3A_2A_1A_0$ et $B_3B_2B_1B_0$), et donne une réponse dans un mot de 4 bits aussi ($S_3S_2S_1S_0$). (Un ordinateur moderne travaille en réalité sur des mots de 64 bits).
   Schéma issu de wikipédia :



   Remarques : il n'y a pas de retenue entrant sur le 1er additionneur (on pourrait utiliser un demi-additionneur) et la retenue de sortie du dernier additionneur est perdue puisque le résultat est stockées dans un mot de 4 bits seulement (comme les entrées).

   i. Câbler un additionneur 4 bits. On pourra utiliser le module altfullfadder fourni dans la bibliothèque.

   ii. Tester son fonctionnement. Réaliser quelques additions simples.

   iii. Tester l'addition de 5 + 13. Que penser du résultat ? Justifier.