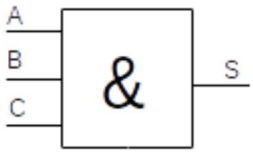
Exercice 1

A partir de l'expression `Y = not (A or B)` , retrouver "tout le reste" : nom, symbole, table de vérité

Exercice 2



A partir du symbole ci-dessus, retrouver "tout le reste" : nom, expression booléenne, table de vérité

Exercice 3

1. Sur feuille, calculer le résultat des expressions suivantes :
   - `S1 = True and False`
   - `S2 = False or not(False)`
   - `S3 = not(True and (False or True))`
   - `S4 = True and (False or True)`
   - `S5 =  not(True) or not((False or True))`

1. Vérifiez vos résultat à l'aide de Thonny

Exercice 4

1. On donne a = 1, b = 0, c = 1. Donner le résultat des expressions booléennes suivantes :
   - `S6 = not(a) xor b`
   - `S7 = a xor b xor c`
   - `S8 = ((a or b) and c) xor c`
   - `S9 = not(a) and (b or not(c))`
   - `S10 = a or b and c`

1. Vérifiez vos résultats à l'aide de Thonny

1. Ecrire la table de vérité des expressions booléennes ci-dessus.
   (Conseil : Quand l'expression booléenne se complique, notamment pour S8, ajouter des colonnes dans la table de vérité permettant de faire des calculs intermédiaires.)

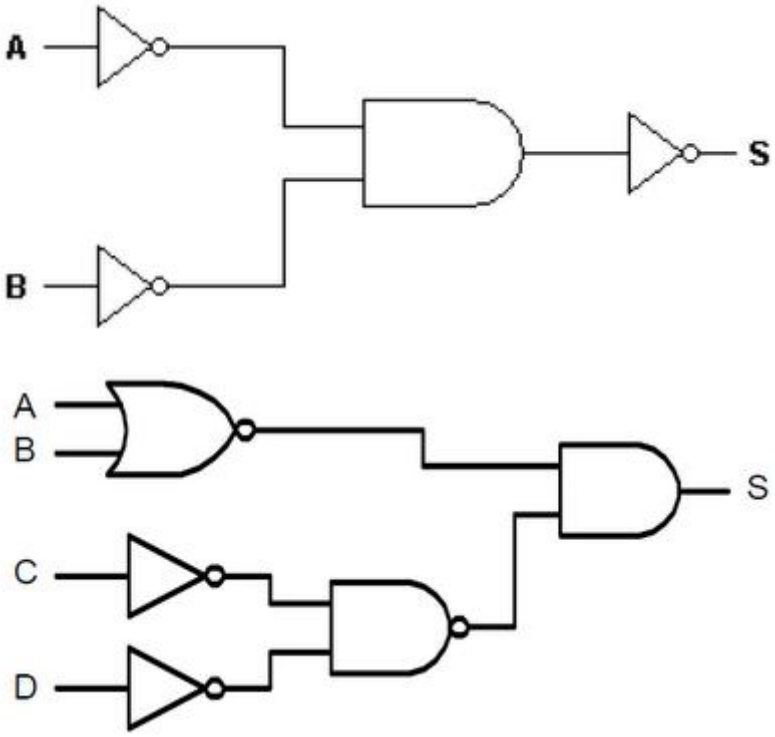
Exercice 5

Simplifier les expressions booléennes suivantes :

- `S11 = x or x`
- `S12 = e or not(e)`
- `S13 = a and not(a)`
- `S14 = y and (not(y) + x)`
- `S15 = a or (not(a) and b)`

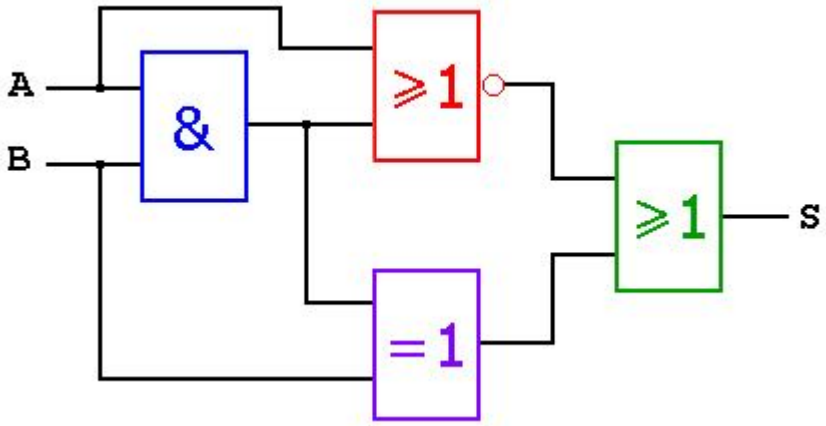
Exercice 6

Pour chaque circuit combinatoire ci-dessous, donner l'équation booléenne de la sortie en fonction des entrées



Exercice 7

1. Donner la table de vérité correspondant au circuit logique ci-dessous
   (Conseil : Ajouter des colonnes dans la table de vérité permettant de faire des calculs intermédiaires.)



1. A partir de la table de vérité, donner sans calcul, l'expression booléenne simplifiée de S

Exercice 8

Pour chaque expression booléenne ci-dessous, dessiner le circuit combinatoire correspondant.

- En norme US : `S6 = not(a) xor b`
- En norme internationale : `S8 = ((a or b) and c) xor b`
- En norme US : `S9 = not(a) and (b or not(c))`
- En norme internationale : `S10 = a or b and c`

Exercice 9

> L'opérateur `xor` n'existe pas en python. Nous allons donc y remédier en créant une fonction

Ecrire en python, une fonction `xor` réalisant l'opération du même nom. Cette fonction :

- prend deux **paramètres** de type **booléen**.
- **Renvoi**e un résultat de type **booléen**.

Remarque : Vous pouvez écrire 2 versions de cette fonction `xor` :

- Version 1 : En utilisant une structure conditionnelle (`if`)
- Version 2 (qui est meilleure) : En utilisant une seule expression booléenne (sans `if`). Pour cela, compléter d'abord l'égalité ci-dessous en observant la table de vérité de la porte logique `xor`

`a xor b = (not(a) and b) or ....`

Exercice 10

Remarque :

- Si nécessaire revoir le cours sur la représentation des entiers positifs et le TD numération dans le bloc 1.
- attention lorsque vous vérifiez vos résultats sous thonny, python renvoie systématiquement le résultat en forme décimale. Vous pouvez utiliser les fonctions bin() et hex() pour convertir votre résultat dans le base souhaitée

1. Sans l'aide de l'ordinateur, calcuer le résultat des expressions python ci-dessous :
   - `0b1010 << 3`
   - `45 >> 1`
   - `0b110101 & 0b1100`
   - `0xE445 & 0x00FF`
   - `0b110101 | 0b1100`
   - `0xE445 | 0x00FF`
   - `0b110101 ^ 0b1100`
   - `85 ^ 47`

1. Vérifiez vos résultats à l'aide de Thonny

Exercice 11 : niveau facile

> On considère trois nombres a, b et c. On considère que ces trois nombres sont classés par ordre croissant si l'inégalité $a < b < c$ est vérifiée. La fonction `croissant` ci-dessous renvoie `True` si

In [2]:
```
def croissant(a,b,c):
    if a <= b :
        if b <= c :
            resultat = True
        else :
            resultat = False
    else :
        resultat = False
    return resultat
```

In [3]:
```
croissant (2,6,9)
```

Out[3]: `True`

```
In [4]: croissant (2,6,5)
Out[4]: False
```

Cette fonction étant un **prédicat**, compléter ci-dessous le code cette fonction en éliminant le `if` de son code

```
In [ ]: def croissant(a,b,c):
            return ......
```

Exercice 12 : niveau facile

(D'après Fil (https://www.fil.univ-lille1.fr))

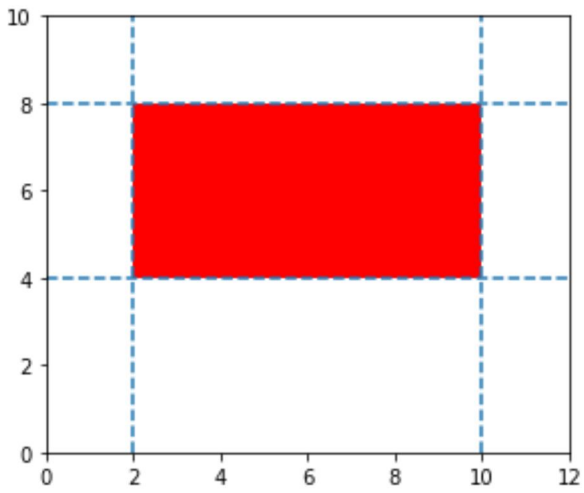
1. Ecrire une fonction **prédicat** `est_positif` renvoyant `True` si le nombre passé en paramètre est positif ou nul, et `False` s'il est strictement négatif.

1. Ecrire une fonction **prédicat** `meme_signe` renvoyant `True` si les deux nombres passés en paramètre sont soit tous les deux positifs ou nuls, soit tous les deux négatifs ou nuls, et `False` sinon

Exercice 13 : niveau intermédiaire

(D'après Fil (https://www.fil.univ-lille1.fr))

Ecrire une fonction **prédicat** `dans_rectangle` prenant en **paramètres** deux nombres représentant l'abscisse et l'ordonnée d'un point M et renvoyant `True` si M est dans le rectangle rouge et `False` sinon.



Exercice 14 : niveau difficile

On donne la fonction `est_bissextile` suivante qui **renvoie un booléen égal à `True`** si et seulement si l'année (type `int`) passée en paramètre est bissextile. Remarque : une année bissextile n'est pas juste une année divisible par 4, c'est un peu plus compliqué que cela et c'est ce que réalise cette fonction.

In [1]:
```
def est_bissextile(annee):
    if annee % 4 == 0:
            if annee % 100 == 0:
                    if annee % 400 == 0:
                            return True
                    else:
                            return False
            else:
                    return True
    else:
            return False
```

In [12]:
```
est_bissextile(2000)
```

Out[12]:
```
False
```

In [10]:
```
est_bissextile(1000)
```

Out[10]:
```
False
```

Cette fonction étant un **prédicat**, compléter ci-dessous le code cette fonction en éliminant le `if` de son code

In [9]:
```
def est_bissextile(annee):
    return .....
```