Ex n°1:

a	not(b)	a or (not(b))
0	0	0
0	1	1
1	0	1
1	1	1

Ex n°2:

not(a)	b	(not(a)) and b
0	0	0
0	1	0
1	0	0
1	1	1

Ex n°3:

a	b	c	(a or b) and c
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Ex n°4:

a	b	c	(a and b) or c
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Ex n°5:

a	b	(a or b) and (not(a and b))
0	0	0
0	1	1
1	0	1
1	1	0

a xor b est équivalent à (a ou b) et (non (a et b)).

Ex n°6:

```
print("----------------------Ex n°1----------------------")
def operateur_et(a,b):
    if a == 1:
        if b == 1:
            return(1)
        elif b == 0:
            return(0)
```

```
    elif a == 0:
        if b == 1:
            return(0)
        elif b == 0:
            return(0)

print("operateur-et test1 reussi", operateur_et(0,0) == 0)
print("operateur-et test2 reussi", operateur_et(0,1) == 0)
print("operateur-et test3 reussi", operateur_et(1,0) == 0)
print("operateur-et test4 reussi", operateur_et(1,1) == 1)
```

Ex n°6:

```
print("----------------------Ex n°2----------------------")
def operateur_ou(a,b):
    if a == 1:
        if b == 1:
            return(1)
        elif b == 0:
            return(1)
    elif a == 0:
        if b == 1:
            return(1)
        elif b == 0:
            return(0)

print("operateur-ou test1 reussi", operateur_ou(0,0) == 0)
print("operateur-ou test2 reussi", operateur_ou(0,1) == 1)
print("operateur-ou test3 reussi", operateur_ou(1,0) == 1)
print("operateur-ou test4 reussi", operateur_ou(1,1) == 1)
```