Lycée Les 3 Sources — **Bourg-Lès-Valence**
NSI 1re — Année **2025-26**

Devoir surveillé n°2 – correction

Représentation des données : types construits

Exercice 1 : (2,5pt) On considère les instructions suivantes :

```
>>> liste1 = [3, 7, 5, 1, 2, 4, 0]
>>> liste2 = [5, 6]
>>> liste3 = [0, 9, 4]
```

1) Que vaut chacune des expressions suivantes :
   a) `liste1[1]` : 7
   b) `liste3 + liste2` : `[0, 9, 4, 5, 6]`
   c) `liste2[1] + liste1[2]` : 11 (on fait la somme de 6 et de 5)
   d) `liste1[liste1[4]]` : 5
2) On rajoute les instructions suivantes :

```
>>> liste4 = [5, 2, 1, 8]
>>> liste4.append(7)
>>> liste4.append(liste3[0])
```

   Quelle est alors la valeur de `liste4` ? `[5, 2, 1, 8, 7, 0]`

Exercice 2 : (2pt) On rappelle que `range(N)` renvoie successivement toutes les valeurs entières allant de `0` à `N-1` inclus.
Déterminer les listes obtenues à l'aide des expressions suivantes :

1) `[2*i for i in range(5)]` : `[0, 2, 4, 6, 8]`
   (on multiplie par 2 chaque élément de `[0, 1, 2, 3, 4]`)
2) `[i for i in range(10) if i >= 5]` : `[5, 6, 7, 8, 9]`
   (on ne garde que les éléments supérieurs ou égaux à 5)

Algorithmique

Exercice 3 : (2pt) Écrire le code de la fonction `indice` qui prend en paramètre une valeur `val` et une liste `valeurs` et qui renvoie l'indice de la première occurrence de `val` dans `valeurs`, s'il y en a une, et `-1` sinon. Vous ne pouvez pas utiliser la fonction `index`.

```
def indice(val, valeurs):
    for i in range(len(valeurs)):
        if val == valeurs[i]:
            return i
    return -1
```

```
>>> indice(4, [3, 4, -4, 6, 4])
1
>>> indice(6, [3, 4, -4, 6, 4])
3
```

```
>>> indice(-4, [3, 4, -4, 6, 4])
2
>>> indice(7, [3, 4, -4, 6, 4])
-1
```

Exercice 4 : (2pt) Écrire le code de la fonction `maximum` qui prend en paramètre une liste `nombres` et qui renvoie la plus grande valeur contenue dans `nombres`. On suppose que la liste n'est pas vide. Vous ne pouvez pas utiliser la fonction **`max`**.

```
def maximum(nombres):
    maxi = nombres[0]
    for v in nombres:
        if v > maxi:
            maxi = v
    return maxi
```

```
>>> maximum([5, 1, 8, 14, 2])
14
>>> maximum([15])
15
```

```
>>> maximum([-9, -27, -2, -48])
-2
>>> maximum([9, 2, -57, 1, 8])
9
```

Exercice 5 : (3pt) On considère la fonction ci-dessous.

```
def inconnu(liste1, liste2):
    nouvelle_liste = []
    for i in range(len(liste1)):
        if liste1[i] > liste2[i]:
            nouvelle_liste.append(liste1[i])
        else:
            nouvelle_liste.append(liste2[i])
        # completer une ligne du tableau
    return nouvelle_liste
```

1) Compléter le tableau ci-contre pour l'exécution de `inconnu([5, 4, 2, 7], [6, 7, 1, 4])`. Entourer la valeur renvoyée à la fin.

i	liste1[i]	liste2[i]	nouvelle_liste
			[]
0	5	6	[6]
1	4	7	[6, 7]
2	2	1	[6, 7, 2]
3	7	4	[6, 7, 2, 7]

2) Quelle est la valeur renvoyée pour `inconnu([3, 2, 6, 7, 9, 3], [3, 1, 7, 6, 9, 4])` ?
`[3, 2, 7, 7, 9, 4]`

3) Décrire en français ce que fait cette fonction. Par exemple : "elle fait la somme de tous les éléments de la liste". On suppose que `liste1` et `liste2` ont la même longueur.
Cette fonction renvoie une liste de même longueur que les deux autres en prenant à chaque fois le nombre les plus grand entre les deux éléments de même indice.

Exercice type bac

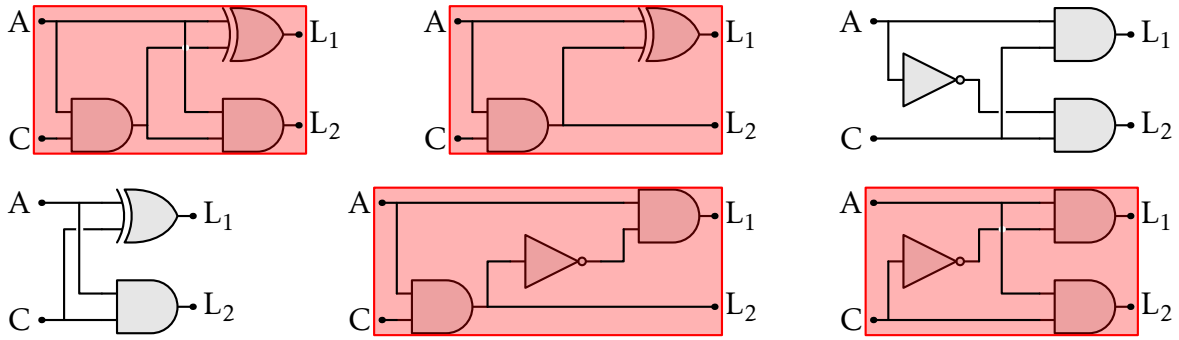
Exercice 6 : (18,5pt) Une commerçante souhaite fabriquer un petit dispositif à mettre dans sa vitrine afin d'attirer le regard. Elle pense à un système avec des LEDs qui clignotent alternativement. Pour simplifier, on va supposer qu'il n'y a que 4 LEDs. Soit la 1re et la 3e sont allumées, soit ce sont les deux autres.

Partie A : Circuits logiques

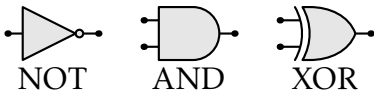
La première idée de la commerçante, qui aime bien bricoler, c'est de faire un circuit. Elle veut 2 entrées : A et C. A est un interrupteur. Lorsqu'il est allumé, le dispositif est en marche. C est une horloge et selon si elle est à 0 ou à 1, c'est soit la LED L_1, soit la LED L_2 qui est allumée. Cela correspond à la table de vérité ci-contre.

A	C	L_1	L_2
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Elle hésite entre plusieurs circuits.



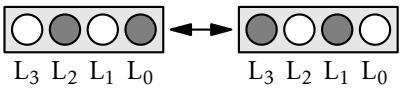
1) Entourer tous les circuits qui correspondent à la table.
Il y a des copies de ces circuits à la page 6 qui peuvent vous servir de brouillon.



Partie B : Nombres binaires

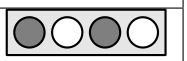
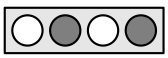
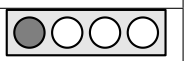
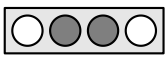
Finalement, elle trouve que les circuits logiques ne donnent pas assez de flexibilité. Si elle veut rajouter des LEDs ou modifier le motif, elle devra changer le circuit. Elle préfère utiliser un processeur programmable afin d'avoir plus de flexibilité.
Pour préparer le programme, elle a besoin de travailler quelques notions de binaire.

Tout d'abord, on note les LEDs L_3, L_2, L_1 et L_0, et on les représente comme si contre. **Les lampes allumées sont en blanc et celles éteintes sont en gris.**



On associe l'état des LEDs à un nombre binaire. Le bit de gauche, celui correspondant à la plus haute puissance de 2, est associé à L_3, alors que celui de droite, qui correspond à la plus petite puissance de 2, est associé à L_0. Par exemple 1101_2 correspond à  . Ces nombres sont ensuite écrits en base 10. L'exemple précédent s'écrit donc 13.

2) Pour les lignes 2 et 3 du tableau ci-contre, écrire le code, c'est-à-dire la valeur décimale, correspondant aux lampes allumées.
3) Pour les lignes 4 et 5, colorier les lampes éteintes pour obtenir le motif correspondant au code donné.

Code	Lampes
13	
5	
10	
7	
9	

Au lieu de faire un simple clignotement, elle envisage plutôt de faire un défilement des lampes.

4) Par quel nombre, en base 10, faut-il multiplier un nombre pour décaler tous ses chiffres en binaire d'un rang vers la gauche ? Par exemple, pour passer de 1101_2 à 11010_2. Par 2

Le problème avec cette méthode, c'est qu'au bout d'un moment, le nombre binaire finit par avoir plus de 4 bits comme dans l'exemple de la question précédente. Pour régler cela, elle souhaite utiliser le ET bit à bit entre deux nombres binaires.

On applique l'opération booléenne ET entre le dernier bit du premier nombre et celui du deuxième, puis le ET entre l'avant-dernier bit de chacun des deux nombres et ainsi de suite. Il n'y a pas de retenues, contrairement à une addition.

```
  1 0 0 0 1 1 0 1
ET 0 1 1 0 1 0 1 1
-----------------
  0 0 0 0 1 0 0 1
```

5) Effectuer l'opération ci-contre.

```
  0 1 0 1 1 0 1 1
ET 0 0 0 1 0 1 1 0
-----------------
  0 0 0 1 0 0 1 0
```

Afin de ne garder que les 4 derniers bits d'un nombre binaire, il faut faire le ET avec $1111_2 = 15$. Si le deuxième nombre a plus de 4 bits, on rajoute des 0 à gauche de 1111_2.

6) Effectuer les opérations ci-contre.

```
  0 0 0 1 0 1 1 0
ET 0 0 0 0 1 1 1 1
-----------------
  0 0 0 0 0 1 1 0
```

```
  0 0 0 0 1 0 1 1
ET 0 0 0 0 1 1 1 1
-----------------
  0 0 0 0 1 0 1 1
```

On peut remarquer que pour certains nombres A, $\boxed{\text{A ET 1111} = \text{A}}$ alors que pour d'autres nombres B, $\boxed{\text{B ET 1111} \neq \text{B}}$.

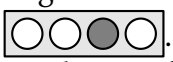
7) Compléter les opérations ci-dessous avec le plus grand A tel que $\boxed{\text{A ET 1111} = \text{A}}$ et le plus petit B tel que $\boxed{\text{B ET 1111} \neq \text{B}}$.

```
  0 0 0 0 1 1 1 1  = A
ET 0 0 0 0 1 1 1 1
-----------------
  0 0 0 0 1 1 1 1  = A
```

```
  0 0 0 1 0 0 0 0  = B
ET 0 0 0 0 1 1 1 1
-----------------
  0 0 0 0 0 0 0 0  ≠ B
```

8) Donner la valeur décimale des nombres A et B trouvés à la question précédente.
A = 15 et B = 16.

Partie C : Assembleur

Elle construit son appareil avec un processeur programmable utilisant un jeu d'instructions de type ARM et des nombres codés sur 8 bits. Pour simplifier, on supposera que le contenu du registre `R10` commande l'affichage des LEDs. Par exemple, si on place 13 dans le registre `R10`, alors les lampes afficheront  .
Voici le programme qu'elle a écrit et la signification des instructions utilisées.

```
1          MOV     R0, #5
2  debut   ADD     R1, R0, R0
3          AND     R0, R1, #15
4          CMP     R0, R1
5          BEQ     affich
6          ADD     R0, R0, #1
7  affich  MOV     R10, R0
8          B       debut
```

Assembleur	Signification
`MOV dest, op1`	dest = op1
`ADD dest, op1, op2`	dest = op1 + op2
`AND dest, op1, op2`	dest = op1 ET op2 Fait le ET bit à bit entre op1 et op2
`CMP op1, op2` `BEQ label`	Aller à l'instruction `label` si op1 = op2
`B label`	Aller à `label`

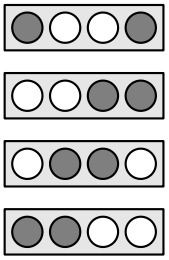
Le programme contient une boucle infinie qui est arrêtée lorsque l'appareil est éteint.

Nom et prénom :

9) Remplir le tableau ci-contre correspondant aux deux premiers tours de boucle du programme. Laissez les cases vides lorsqu'un registre n'a pas encore de valeur (comme `R1` et `R10` au début. Vous pouvez aussi laisser vide la ligne 6 si elle n'est pas exécutée dans un tour de boucle. Vous pouvez remplir les cases en binaire et/ou en décimal.

10) Remplir de la même manière le tableau ci-contre, mais en remplaçant la première ligne par `MOV, R0, #3`.

11) Compléter les lampes ci-dessous qui correspondent au registre `R10` à la fin de chacun des tours de boucle.



ligne	R0	R1	R10
1	5 = 0101		
2	5 = 0101	10 = 1010	
3	10 = 1010	10 = 1010	
6	10 = 1010	10 = 1010	
7	10 = 1010	10 = 1010	10 = 1010
2	10 = 1010	20 = 10100	10 = 1010
3	4 = 0100	20 = 10100	10 = 1010
6	5 = 0101	20 = 10100	10 = 1010
7	5 = 0101	20 = 10100	5 = 0101

ligne	R0	R1	R10
1	3 = 0011		
2	3 = 0011	6 = 0110	
3	6 = 0110	6 = 0110	
6	6 = 0110	6 = 0110	
7	6 = 0110	6 = 0110	6 = 0110
2	6 = 0110	6 = 0110	6 = 0110
3	12 = 1100	12 = 1100	6 = 0110
6	12 = 1100	12 = 1100	6 = 0110
7	12 = 1100	12 = 1100	12 = 1100
2	12 = 1100	12 = 1100	12 = 1100
3	8 = 1000	24 = 11000	12 = 1100
6	9 = 1001	24 = 11000	12 = 1100
7	9 = 1001	24 = 11000	9 = 1001
2	9 = 1001	24 = 11000	9 = 1001
3	2 = 0010	18 = 10010	9 = 1001
6	3 = 0011	18 = 10010	9 = 1001
7	3 = 0011	18 = 10010	3 = 0011

12) Finalement, elle décide de rajouter 2 LEDs, pour un total de 6.

a) Quelle ligne doit être modifiée dans le programme pour pouvoir gérer les 6 lampes et quelle modification doit-elle faire? Il faut mettre `AND R0, R1,#63` à la ligne 3.

b) Déterminer les valeurs, en base 10, prises successivement par `R10` si on remplace la première ligne par `MOV, R0, #18`. 18, 36, 9, 18

Lycée Les 3 Sources — **Bourg-Lès-Valence**
NSI 1re — Année **2025-26**

Devoir surveillé n°2 – correction

Représentation des données : types construits

Exercice 1 : (2,5pt) On considère les instructions suivantes :

```
>>> liste1 = [3, 7, 5, 1, 2, 4, 0]
>>> liste2 = [5, 6]
>>> liste3 = [0, 9, 4]
```

1) Que vaut chacune des expressions suivantes :
   a) `liste1[2]` : 5
   b) `liste2 + liste3` : `[5, 6, 0, 9, 4]`
   c) `liste3[2] + liste1[1]` : 11 (on fait la somme de 4 et de 7)
   d) `liste1[liste1[4]]` : 5
2) On rajoute les instructions suivantes :

```
>>> liste4 = [5, 2, 1, 8]
>>> liste4.append(7)
>>> liste4.append(liste3[0])
```

   Quelle est alors la valeur de `liste4` ? `[5, 2, 1, 8, 7, 0]`

Exercice 2 : (2pt) On rappelle que `range(N)` renvoie successivement toutes les valeurs entières allant de `0` à `N-1` inclus.
Déterminer les listes obtenues à l'aide des expressions suivantes :

1) `[i+3 for i in range(5)]` : `[3, 4, 5, 6, 7]`
   (on ajoute 3 à chaque élément de `[0, 1, 2, 3, 4]`)
2) `[i for i in range(10) if i >= 6]` : `[6, 7, 8, 9]`
   (on ne garde que les éléments strictement supérieurs à 5)

Algorithmique

Exercice 3 : (2pt) Écrire le code de la fonction `indice` qui prend en paramètre une valeur `val` et une liste `valeurs` et qui renvoie l'indice de la première occurrence de `val` dans `valeurs`, s'il y en a une, et `-1` sinon. Vous ne pouvez pas utiliser la fonction `index`.

```
def indice(val, valeurs):
    for i in range(len(valeurs)):
        if val == valeurs[i]:
            return i
    return -1
```

```
>>> indice(4, [3, 4, -4, 6, 4])
1
>>> indice(6, [3, 4, -4, 6, 4])
3
```

```
>>> indice(-4, [3, 4, -4, 6, 4])
2
>>> indice(7, [3, 4, -4, 6, 4])
-1
```

Exercice 4 : (2pt) Écrire le code de la fonction `maximum` qui prend en paramètre une liste `nombres` et qui renvoie la plus grande valeur contenue dans `nombres`. On suppose que la liste n'est pas vide. Vous ne pouvez pas utiliser la fonction **`max`**.

```
def maximum(nombres):
    maxi = nombres[0]
    for v in nombres:
        if v > maxi:
            maxi = v
    return maxi
```

```
>>> maximum([5, 1, 8, 14, 2])
14
>>> maximum([15])
15
```

```
>>> maximum([-9, -27, -2, -48])
-2
>>> maximum([9, 2, -57, 1, 8])
9
```

Exercice 5 : (3pt) On considère la fonction ci-dessous.

```
def inconnu(liste1, liste2):
    nouvelle_liste = []
    for i in range(len(liste1)):
        if liste1[i] > liste2[i]:
            nouvelle_liste.append(liste1[i])
        else:
            nouvelle_liste.append(liste2[i])
        # completer une ligne du tableau
    return nouvelle_liste
```

1) Compléter le tableau ci-contre pour l'exécution de `inconnu([5, 4, 1, 3], [4, 6, 7, 1])`. Entourer la valeur renvoyée à la fin.

i	liste1[i]	liste2[i]	nouvelle_liste
			[]
0	5	4	[5]
1	4	6	[5, 6]
2	1	7	[5, 6, 7]
3	3	1	[5, 6, 7, 3]

2) Quelle est la valeur renvoyée pour `inconnu([3, 1, 6, 3, 2, 9], [3, 7, 6, 1, 4, 9])` ?
`[3, 7, 6, 3, 4, 9]`

3) Décrire en français ce que fait cette fonction. Par exemple : "elle fait la somme de tous les éléments de la liste". On suppose que `liste1` et `liste2` ont la même longueur.
Cette fonction renvoie une liste de même longueur que les deux autres en prenant à chaque fois le nombre les plus grand entre les deux éléments de même indice.

Exercice type bac

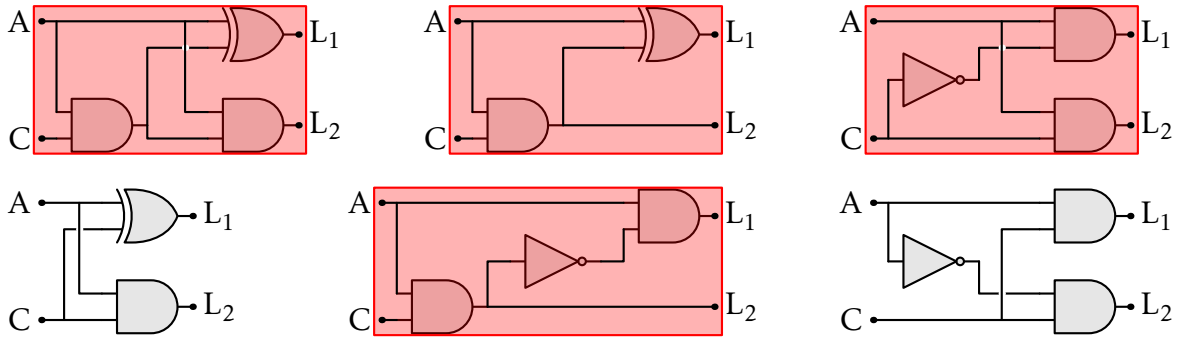
Exercice 6 : (18,5pt) Une commerçante souhaite fabriquer un petit dispositif à mettre dans sa vitrine afin d'attirer le regard. Elle pense à un système avec des LEDs qui clignotent alternativement. Pour simplifier, on va supposer qu'il n'y a que 4 LEDs. Soit la 1re et la 3e sont allumées, soit ce sont les deux autres.

Partie A : Circuits logiques

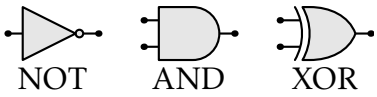
La première idée de la commerçante, qui aime bien bricoler, c'est de faire un circuit. Elle veut 2 entrées : A et C. A est un interrupteur. Lorsqu'il est allumé, le dispositif est en marche. C est une horloge et selon si elle est à 0 ou à 1, c'est soit la LED L_1, soit la LED L_2 qui est allumée. Cela correspond à la table de vérité ci-contre.

A	C	L_1	L_2
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Elle hésite entre plusieurs circuits.



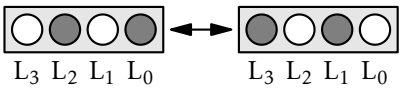
1) Entourer tous les circuits qui correspondent à la table.
Il y a des copies de ces circuits à la page 6 qui peuvent vous servir de brouillon.



Partie B : Nombres binaires

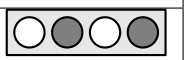
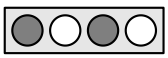
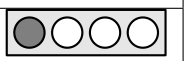
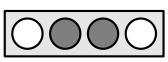
Finalement, elle trouve que les circuits logiques ne donnent pas assez de flexibilité. Si elle veut rajouter des LEDs ou modifier le motif, elle devra changer le circuit. Elle préfère utiliser un processeur programmable afin d'avoir plus de flexibilité.
Pour préparer le programme, elle a besoin de travailler quelques notions de binaire.

Tout d'abord, on note les LEDs L_3, L_2, L_1 et L_0, et on les représente comme si contre. **Les lampes allumées sont en blanc et celles éteintes sont en gris.**



On associe l'état des LEDs à un nombre binaire. Le bit de gauche, celui correspondant à la plus haute puissance de 2, est associé à L_3, alors que celui de droite, qui correspond à la plus petite puissance de 2, est associé à L_0. Par exemple 1101_2 correspond à  . Ces nombres sont ensuite écrits en base 10. L'exemple précédent s'écrit donc 13.

2) Pour les lignes 2 et 3 du tableau ci-contre, écrire le code, c'est-à-dire la valeur décimale, correspondant aux lampes allumées.
3) Pour les lignes 4 et 5, colorier les lampes éteintes pour obtenir le motif correspondant au code donné.

Code	Lampes
13	
10	
5	
7	
9	

Au lieu de faire un simple clignotement, elle envisage plutôt de faire un défilement des lampes.

4) Par quel nombre, en base 10, faut-il multiplier un nombre pour décaler tous ses chiffres en binaire d'un rang vers la gauche ? Par exemple, pour passer de 1101_2 à 11010_2. Par 2

Le problème avec cette méthode, c'est qu'au bout d'un moment, le nombre binaire finit par avoir plus de 4 bits comme dans l'exemple de la question précédente. Pour régler cela, elle souhaite utiliser le ET bit à bit entre deux nombres binaires.

On applique l'opération booléenne ET entre le dernier bit du premier nombre et celui du deuxième, puis le ET entre l'avant-dernier bit de chacun des deux nombres et ainsi de suite. Il n'y a pas de retenues, contrairement à une addition.

```
  1 0 0 0 1 1 0 1
ET 0 1 1 0 1 0 1 1
-----------------
  0 0 0 0 1 0 0 1
```

5) Effectuer l'opération ci-contre.

```
  0 1 1 0 0 1 0 0
ET 0 1 0 1 0 1 1 0
-----------------
  0 1 0 0 0 1 0 0
```

Afin de ne garder que les 4 derniers bits d'un nombre binaire, il faut faire le ET avec $1111_2 = 15$. Si le deuxième nombre a plus de 4 bits, on rajoute des 0 à gauche de 1111_2.

6) Effectuer les opérations ci-contre.

```
  0 0 0 1 0 0 1 1
ET 0 0 0 0 1 1 1 1
-----------------
  0 0 0 0 0 0 1 1
```

```
  0 0 0 0 1 1 1 0
ET 0 0 0 0 1 1 1 1
-----------------
  0 0 0 0 1 1 1 0
```

On peut remarquer que pour certains nombres A, $\boxed{\text{A ET 1111} = \text{A}}$ alors que pour d'autres nombres B, $\boxed{\text{B ET 1111} \neq \text{B}}$.

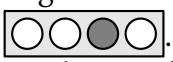
7) Compléter les opérations ci-dessous avec le plus grand A tel que $\boxed{\text{A ET 1111} = \text{A}}$ et le plus petit B tel que $\boxed{\text{B ET 1111} \neq \text{B}}$.

```
  0 0 0 0 1 1 1 1  = A
ET 0 0 0 0 1 1 1 1
-----------------
  0 0 0 0 1 1 1 1  = A
```

```
  0 0 0 1 0 0 0 0  = B
ET 0 0 0 0 1 1 1 1
-----------------
  0 0 0 0 0 0 0 0  ≠ B
```

8) Donner la valeur décimale des nombres A et B trouvés à la question précédente.
A = 15 et B = 16.

Partie C : Assembleur

Elle construit son appareil avec un processeur programmable utilisant un jeu d'instructions de type ARM et des nombres codés sur 8 bits. Pour simplifier, on supposera que le contenu du registre `R10` commande l'affichage des LEDs. Par exemple, si on place 13 dans le registre `R10`, alors les lampes afficheront  .
Voici le programme qu'elle a écrit et la signification des instructions utilisées.

```
1          MOV     R0, #5
2  debut   ADD     R1, R0, R0
3          AND     R0, R1, #15
4          CMP     R0, R1
5          BEQ     affich
6          ADD     R0, R0, #1
7  affich  MOV     R10, R0
8          B       debut
```

Assembleur	Signification
`MOV dest, op1`	dest = op1
`ADD dest, op1, op2`	dest = op1 + op2
`AND dest, op1, op2`	dest = op1 ET op2 Fait le ET bit à bit entre op1 et op2
`CMP op1, op2` `BEQ label`	Aller à l'instruction `label` si op1 = op2
`B label`	Aller à `label`

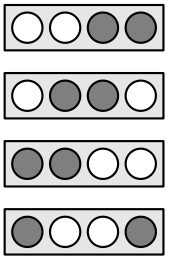
Le programme contient une boucle infinie qui est arrêtée lorsque l'appareil est éteint.

Nom et prénom :

9) Remplir le tableau ci-contre correspondant aux deux premiers tours de boucle du programme. Laissez les cases vides lorsqu'un registre n'a pas encore de valeur (comme `R1` et `R10` au début. Vous pouvez aussi laisser vide la ligne 6 si elle n'est pas exécutée dans un tour de boucle. Vous pouvez remplir les cases en binaire et/ou en décimal.

10) Remplir de la même manière le tableau ci-contre, mais en remplaçant la première ligne par `MOV, R0, #6`.

11) Compléter les lampes ci-dessous qui correspondent au registre `R10` à la fin de chacun des tours de boucle.



ligne	R0	R1	R10
1	5 = 0101		
2	5 = 0101	10 = 1010	
3	10 = 1010	10 = 1010	
6	10 = 1010	10 = 1010	
7	10 = 1010	10 = 1010	10 = 1010
2	10 = 1010	20 = 10100	10 = 1010
3	4 = 0100	20 = 10100	10 = 1010
6	5 = 0101	20 = 10100	10 = 1010
7	5 = 0101	20 = 10100	5 = 0101

ligne	R0	R1	R10
1	6 = 0110		
2	6 = 0110	12 = 1100	
3	12 = 1100	12 = 1100	
6	12 = 1100	12 = 1100	
7	12 = 1100	12 = 1100	12 = 1100
2	12 = 1100	12 = 1100	12 = 1100
3	8 = 1000	24 = 11000	12 = 1100
6	9 = 1001	24 = 11000	12 = 1100
7	9 = 1001	24 = 11000	9 = 1001
2	9 = 1001	24 = 11000	9 = 1001
3	2 = 0010	18 = 10010	9 = 1001
6	3 = 0011	18 = 10010	9 = 1001
7	3 = 0011	18 = 10010	3 = 0011
2	3 = 0011	18 = 10010	3 = 0011
3	6 = 0110	6 = 0110	3 = 0011
6	6 = 0110	6 = 0110	3 = 0011
7	6 = 0110	6 = 0110	6 = 0110

12) Finalement, elle décide de rajouter 2 LEDs, pour un total de 6.

a) Quelle ligne doit être modifiée dans le programme pour pouvoir gérer les 6 lampes et quelle modification doit-elle faire? Il faut mettre `AND R0, R1,#63` à la ligne 3.

b) Déterminer les valeurs, en base 10, prises successivement par `R10` si on remplace la première ligne par `MOV, R0, #36`. 36, 9, 18, 36