Contrôle

NOM et PRÉNOM :

Barème sur 29 points.

Calculatrice interdite.

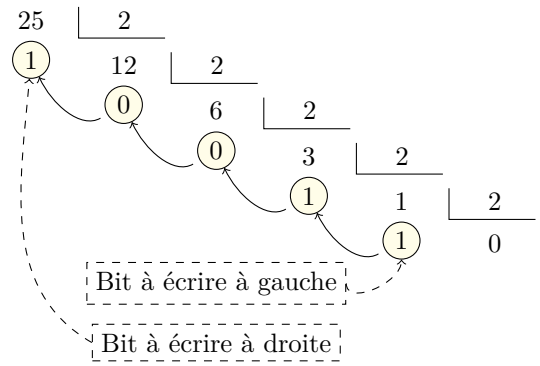
Exercice 1 (2 points, du décimal au binaire) –

1. Donner l'écriture binaire de l'entier $n = 25_{\text{dix}}$. Donner le détail du calcul par une division en cascade.

2. Donner l'écriture binaire de l'entier $n = 10_{\text{dix}}$.

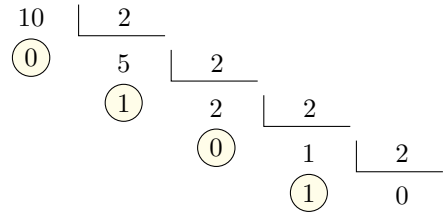
Résolution.

1. Écriture binaire de 25.
   On commence par poser la cascade :



   On a donc : $25_{\text{dix}} = 11001_{\text{deux}}$.

2. Écriture binaire de l'entier 10.
   On pose la cascade :



   On a donc : $10_{\text{dix}} = 1010_{\text{deux}}$.

Exercice 2 (2 points, vers le décimal) –

1. Donner l'écriture décimale de l'entier $m = 101100_{\text{deux}}$.

2. Donner l'écriture décimale de l'entier $q = \text{ac}_{\text{seize}}$.

Résolution.

1. L'écriture décimale de l'entier $m = 101100_{\text{deux}}$.

$$\begin{aligned} 101100_{\text{deux}} &= 2^5 + 2^3 + 2^2 \\ &= 32 + 8 + 4 \\ &= 44 \end{aligned}$$

2. L'écriture décimale de l'entier $q = \text{ac}_{\text{seize}}$.

$$\begin{aligned} \text{ac}_{\text{seize}} &= 10 \times 16 + 12 \\ &= 160 + 12 \\ &= 172 \end{aligned}$$

Exercice 3 (2 points, binaire ↔ hexadécimal) –

1. Donner l'écriture hexadécimale de l'entier $p = 10\,11101100_{\text{deux}}$.

2. Donner l'écriture binaire de l'entier $r = \text{fade2}_{\text{seize}}$.

Résolution.

1. L'écriture hexadécimale de l'entier $p = 10\ 1110\ 1100_{\text{deux}}$.
   On traduit chaque paquet de 4 bits en base seize :

$$p = 2\text{ec}_{\text{seize}}$$

2. L'écriture binaire de l'entier $r = \text{fade2}_{\text{seize}}$.
   On traduit chaque chiffre de la base seize en binaire, sur une longueur de 4 bits :

$$r = 1111\ 1010\ 1101\ 1110\ 0010_{\text{deux}}$$

Exercice 4 (2 points) –
Compléter le corps de la fonction ci-dessous :

```
def ecritureBinaire(n):
    """
    n -- entier naturel (type int).

    renvoie une chaîne de caractères.
    Cette chaîne de caractères
    est constituée de '0' et de '1'
    et elle correspond à l'écriture binaire
    de l'entier n.
    L'algorithme utilisé est celui de la division en cascade.
    >>> ecritureBinaire(4)
    '100'
    >>> ecritureBinaire(18)
    '10010'
    """
    # on traite le cas particulier de 0 à part:
    if n == 0: return '0'

    # les autres cas:
    ch = ''
    while n != 0:
        ..........................
        ............................
    return ch
```

Résolution.

```
def ecritureBinaire(n):
    """
    n -- entier naturel (type int).

    renvoie une chaîne de caractères.
    Cette chaîne de caractères
    est constituée de '0' et de '1'
    et elle correspond à l'écriture binaire
    de l'entier n.
    L'algorithme utilisé est celui de la division en cascade.
    >>> ecritureBinaire(4)
    '100'
    >>> ecritureBinaire(18)
    '10010'
    """
    # on traite le cas particulier de 0 à part:
    if n == 0: return '0'

    # les autres cas:
    ch = ''
```

```
    while n != 0 :
        ch = str(n%2) + ch # écriture du nouveau chiffre à gauche des précédents
        n = n//2 # n devient le quotient suivant
    return ch
```

Exercice 5 (2 points) –
Dans un style css, on a utilisé :

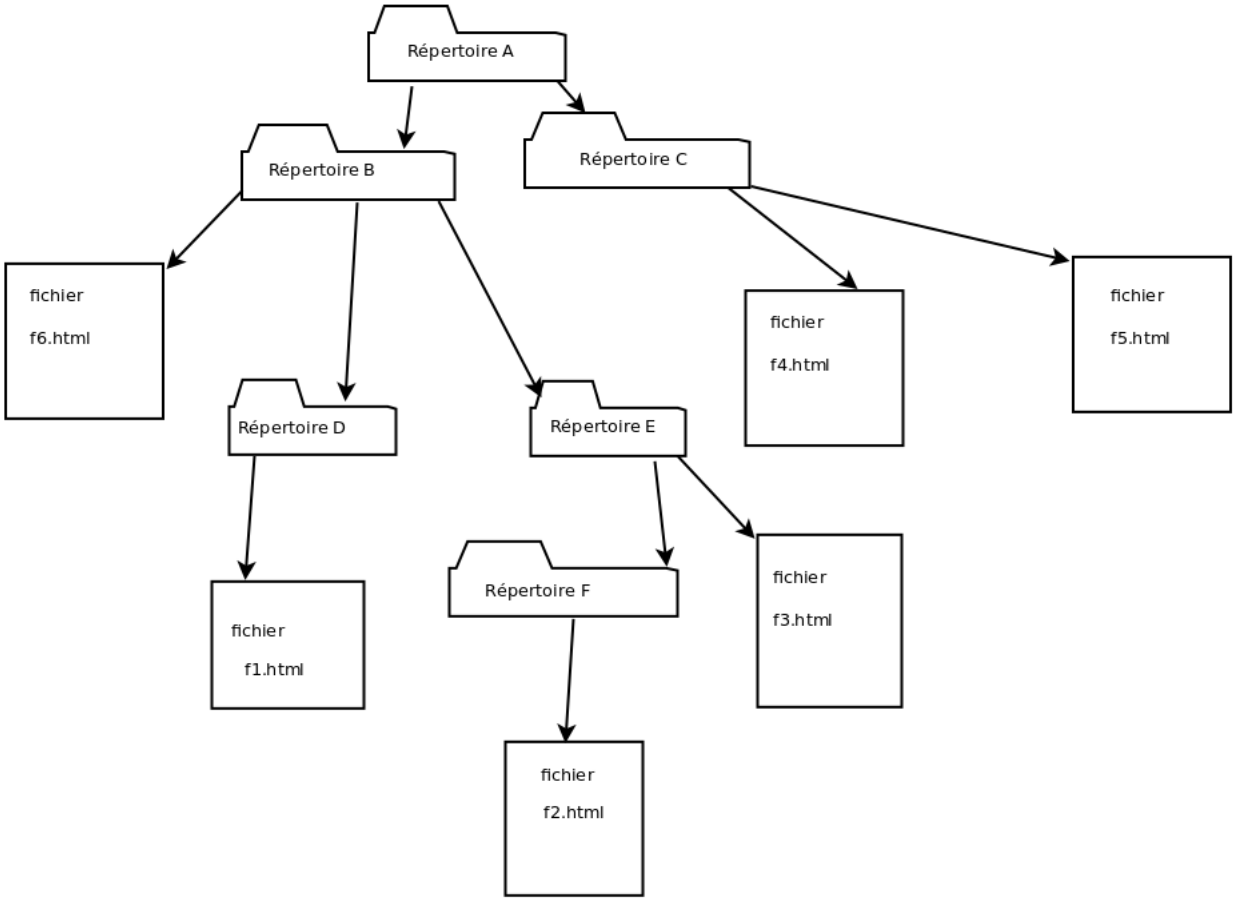
```
div{border-color  : #12aa00; /* couleur de bordure */
    color  : #001a02; /* couleur du texte */
}
```

Réécrire cette règle css en utilisant le codage rgb (décimal) des couleurs.

Résolution.

```
div{
    border-color : rgb(18, 170, 0); /* couleur de bordure */
    color : rgb(0, 26, 2); /* couleur du texte */
}
```

Exercice 6 (4 points) –
Un site web présente l'arborescence suivante :



1. On veut écrire un lien hypertexte dans le fichier f6.html vers le fichier f1.html. Compléter :

```
<a href="....." > lien vers le fichier f1 </a>
```

2. On veut écrire un lien hypertexte dans le fichier f1.html vers le fichier f6.html. Compléter :

```
<a href="....." > lien vers le fichier f6 </a>
```

3. On veut écrire un lien hypertexte dans le fichier f1.html vers le fichier f4.html. Compléter :

```
<a href="....." > lien vers le fichier f4 </a>
```

4. On veut écrire un lien hypertexte dans le fichier f5.html vers le fichier f4.html. Compléter :

```
<a href="....." > lien vers le fichier f4 </a>
```

Résolution.

1. On veut écrire un lien hypertexte dans le fichier f6.html vers le fichier f1.html. Compléter :

```
<a href="D/f1.html"> lien vers le fichier f1 </a>
```

2. On veut écrire un lien hypertexte dans le fichier f1.html vers le fichier f6.html. Compléter :

```
<a href="../f6.html"> lien vers le fichier f6 </a>
```

3. On veut écrire un lien hypertexte dans le fichier f1.html vers le fichier f4.html. Compléter :

```
<a href="../../C/f4.html"> lien vers le fichier f4 </a>
```

4. On veut écrire un lien hypertexte dans le fichier f5.html vers le fichier f4.html. Compléter :

```
<a href="f4.html">  lien vers le fichier f4 </a>
```

Exercice 7 (3 points) –
On définit le tuple suivant :

```
t = (   (1,2,3),
    "a",
    ("python", "C", "javascript", "Go"),
    ("graphe", "logique", "automate", "Algèbre de Boole", "logique floue")
    )
```

1. Quelle est la valeur de t[0] ? .....
2. Quelle est la valeur de t [0][0] ?.....
3. Quelle est la valeur de t[1] ?.....
4. Quelle est la valeur de t [2][0] ?.....
5. Quelle est la valeur de len(t)?.....
6. Quelle est la valeur de len(t[3]) ?.....

Résolution.
1. Valeur de t[0] : (1, 2, 3)
2. Valeur de t [0][0] : 1
3. Valeur de t[1] : 'a'
4. Valeur de t [2][0] : 'python'
5. Valeur de len(t) : 4
6. Valeur de len(t[3]) : 5

Exercice 8 (4 points) –
Compléter le code de la fonction :

```
def somme(t):
    """
    t -- tuple dont les éléments sont des entiers

    renvoie la somme des éléments de t
    """
    ................................
    ................................
    ................................
    ................................
```

Résolution.
Voir `http://iamjmm.ovh/NSI/tuples/site/parcourir.html#exercice_1`

```
def somme(t):
    """
    t -- tuple dont les éléments sont des entiers

    renvoie la somme des éléments de t
    """
```

```
    s = 0
    for entier in t :
        s = s + entier
    return s
```

Exercice 9 (3 points) –
Compléter le code de la fonction :

```
def coincidence(t):
    """
    t -- tuple d'entiers naturels

    renvoie le nombre de valeurs vérifiant : valeur = indice

    >>> coincidence((3, 2, 6))
    0
    >>> coincidence((3, 1, 6))
    1
    >>> coincidence((0, 2, 6, 3))
    2
    """
    nb_coincidence = 0
    ..................................
    ..................................
    ..................................
    return nb_coincidence
```

Résolution.
Voir l'exercice `http://iamjmm.ovh/NSI/tuples/site/parcourir.html#exercice_2`

```
def coincidence(t):
    """
    t -- tuple d'entiers naturels

    renvoie le nombre de valeurs vérifiant : valeur = indice

    >>> coincidence((3, 2, 6))
    0
    >>> coincidence((3, 1, 6))
    1
    >>> coincidence((0, 2, 6, 3))
    2
    """
    nb_coincidence = 0
    for i, v in enumerate(t):
        if i == v:
            nb_coincidence += 1
    return nb_coincidence
```

Exercice 10 (5 points, QCM) –
Pour chaque QCM, entourez la bonne réponse (il y a une et une seule bonne réponse pour chaque question).

QCM 1 – On ajoute un 0 à droite de l'écriture binaire d'un entier n (par exemple, avec $n = 101101111011_{\text{deux}}$, on obtient $1011011110110_{\text{deux}}$). Quelle opération correspond à cela :

a) Multiplier n par l'entier 10_{dix}. b) Multiplier n par l'entier 2_{dix}. c) Multiplier n par l'entier 16_{dix}. d) Ajouter 10_{dix} à n.

Résolution.
b) Multiplication par 2.

QCM 2 – Sur un octet, les entiers naturels que l'on peut écrire en binaire sont :

a) Les entiers de 0 à 2^8. b) Les entiers de 0 à $2^7 - 1$. c) Les entiers de 0 à $2^8 - 1$. d) Les entiers de 0 à 2^7.

Résolution.
Les entiers de 0 à $2^8 - 1$.

QCM 3 – L'écriture décimale de 42_{seize} est :

a) 42_{dix} b) 66_{dix} c) 777_{dix} d) 666_{dix}

Résolution.
66

QCM 4 – Dans un style CSS, on a utilisé :

```
p{background-color : rgb(42, 1,66);}
```

Pour déclarer cette couleur en hexadécimal, on écrira :

a) p{background-color : #420166;} b) p{background-color : rgb(#2a, #1, #42);}
c) p{background-color : #2a0142;} d) p{background-color : #42166;}

Résolution.
p{background-color : #2a0142;}

QCM 5 – Avec le code (dans un terminal) :

```
>>> t = (1, 2, 3)
>>> t[2] = (6,7)
```

a) t vaut maintenant (1,2,(6,7)) b) t vaut maintenant (1,(6,7),3)
c) on obtient une erreur 'tuple' object does not support item assignment
d) On obtient une erreur out of range

Résolution.
'tuple' object does not support item assignment

Exercice 11 (bonus, hors barème) –

Après une course, on dispose d'un tuple constitué de tuples. Chaque tuple est de la forme (numéro de dossard, nom, prénom, (heure, minute, seconde)) où le tuple (heure, minute, seconde) correspond au temps mis par ce candidat pour finir la course.
Par exemple :

```
une_course = ( (1, 'Labrosse', 'Adam', (3,45, 23)),
(2, 'Gemlamorte', 'Adèle', (4, 12, 29)),
(4, 'Etpan', 'Ahmed', (3, 32, 43)),
(5, 'Térieur', 'Alain', (4, 21, 55)),
(3, 'Auboisdormant', 'Abel', (2, 5, 12)),
```

```
(7, 'Tanrien', 'Jean', (3, 42, 10)),
(8, 'Ouzi', 'Jacques', (3, 32, 9)),
(9, 'Deuf', 'John', (5, 40, 7)),
(10, 'Provist', 'Alain', (3,45, 33)),
(6, 'Térieur', 'Alex', (2, 34, 49))
)
```

Compléter le corps de la fonction ci-dessous :

```
def temps_en_seconde(course, dossard):
    """
    course: un tuple tel que le précédent, résultant d'une course.
    dossard: un numéro de dossard
    renvoie le temps en seconde réalisé par le coureur portant ce dossard
    """
    ....................................................
    ....................................................
    ....................................................
    ....................................................
    ....................................................
    ....................................................
```

Résolution.
Voir `http://iamjmm.ovh/NSI/tuples/site/course.html#obtenir_le_temps_en_seconde`

```
def temps_en_seconde(course, dossard):
    """
    course: un tuple tel que le précédent, résultant d'une course.
    dossard: un numéro de dossard
    renvoie le temps en seconde réalisé par le coureur portant ce dossard
    """
    for coureur in course:
        if coureur[0] == dossard:
            heures = coureur[3][0]
            minutes = coureur[3][1]
            secondes = coureur[3][2]
            return heures*3600 + minutes*60 + secondes
```

Exercice 12 (bonus, hors barème) –
Une image contient des pixels blancs et des pixels non blancs. On veut parcourir cette image ligne par ligne (chaque ligne de gauche à droite), en commençant par la ligne du haut pour trouver le premier pixel de couleur distincte de (255, 255, 255).
Les données de l'image ont été "chargées" (comme vu en classe) dans une variable source
(par source = Image.open('nomImage.png')).
Compléter le code ci-dessous. Vous utiliserez la variable 'largeur' et la variable 'hauteur', 'largeur' contenant la largeur de l'image et 'hauteur' contenant la hauteur de l'image en pixels (vous n'avez pas à faire l'affectation de ces deux variables, on suppose que c'est déjà fait dans le reste du script).

```
def premierNonBlanc():
    """
    Parcourt le fichier image ligne par ligne,
    chaque ligne de gauche à droite, en commençant par la ligne du haut.

    Renvoie les coordonnées du premier pixel rencontré
    de couleur distincte de (255, 255, 255).
    """
```

```
    ligne = 0
    colonne = 0

    while source.getpixel((colonne,ligne)) == (255,255,255):

        if ..............:

            ..................
        else:

            ..................

            ..................

    return (colonne, ligne)
```

Résolution.
Sur une ligne, on avance colonne par colonne. Lorsqu'on a traité la dernière colonne (indice largeur-1), on avance d'une ligne et on repart de la colonne 0.

```
def premierNonBlanc():
    """
    Parcourt le fichier image ligne par ligne,
    chaque ligne de gauche à droite, en commençant par la ligne du haut.

    Renvoie les coordonnées du premier pixel rencontré
    de couleur distincte de (255, 255, 255).
    """

    ligne = 0
    colonne = 0

    while source.getpixel((colonne,ligne)) == (255,255,255):

        if colonne < largeur-1:
            colonne += 1
        else:
            colonne = 0
            ligne += 1

    return (colonne, ligne)
```