Épreuves écrites NSI
Corrigé du sujet
Amérique du Nord 2022 Jour 2

Exercice 1

1. (a) On doit compléter le code de la façon suivante :

```
class Concurrent :
    def __init__(self, pseudo, temps, penalite):
        self.nom = pseudo
        self.temps = temps
        self.penalite = penalite
        self.temps_tot = temps + penalite
```

(b) L'attribut aura pour valeur `99.67` soit le résultat de `87.67 + 12`.

(c) On accède à cet attribut par l'expression `c1.temps_tot`.

2. (a) La variable `concurrent` contiendra c4 après exécution des instructions suivantes :

```
    Q = resultats.queue()
    Q = Q.queue()
    concurrent = Q.tete()
```

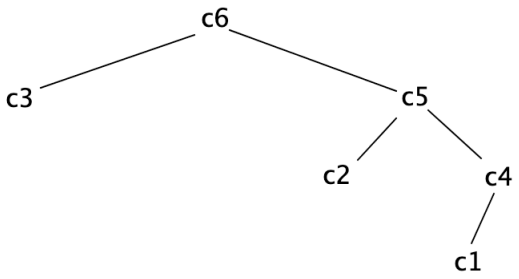
(b) La variable `temps_total` contiendra le temps total du concurrent stocké en tête de la variable `resultats` après exécution de l'instruction suivante :

```
 temps_total = resultats.tete().temps_tot
```

3. On peut compléter le code de la façon suivante :

```
def meilleur_concurrent(L) :
    conc_mini = L.tete()
    mini = conc_mini.temps_tot
    Q = L.queue()
    while not (Q.est_vide()):
        elt = Q.tete()
        if elt.temps_tot < mini :
            conc_mini = elt
            mini = elt.temps_tot
        Q = Q.queue()
    return conc_mini
```

4. On obtient l'arbre binaire suivant :



Exercice 4

1. (a) La valeur de la racine est `"Milka"`.

(b) Les feuilles de l'arbre sont : `"Nemo"`, `"Moka"`, `"Maya"`, `"Museau"` et `"Noisette"`.

(c) "Nuage" est une femelle car il s'agit de la racine du sous-arbre droit de "Nougat".

(d) Le père de "Etoile" est "Ulk" et sa mère est "Maya".

2. (a) On doit compléter le code de la façon suivante :

```
def present(arb, nom):
    if est_vide(arb):
        return False
    elif racine(arb) == nom :
        return True
    else :
        return present(gauche(arb), nom) \
        or present(droit(arb), nom)
```

(b) On peut écrire le code suivant :

```
def parents(arbre):
    if not est_vide(gauche(arbre)):
        pere = racine(gauche(arbre))
    else :
        pere = ''
    if not est_vide(droit(arbre)):
        mere = racine(droit(arbre))
    else :
        mere = ''
    return (pere, mere)
```

3. (a) Milka et Cacao sont frères et sœurs. Mango et Cacao sont frères et sœurs.

(b) On peut écrire le code suivant :

```
def frere_soeur(arbre1, arbre2):
    parents1 = parents(arbre1)
    parents2 = parents(arbre2)
    return parents1[0] == parents2[0] or\
           parents1[1] == parents2[1]
```

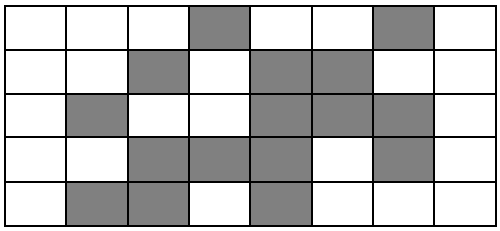
4. On peut écrire le code suivant :

```
def nombre_chiens(arb, n):
    if n == 0 :
        return 1
    else :
        if not est_vide(gauche(arb)) :
            t1 = nombre_chiens(gauche(arb), n-1)
        else :
            t1 = 0
        if not est_vide(droit(arb)) :
            t2 = nombre_chiens(droit(arb), n-1)
        else :
            t2 = 0
    return t1 + t2
```

Exercice 5

Partie A

1. (a) On obtient le dessin suivant :



(b) Le 1er pixel est en ligne 3 et colonne 0.
Le 2ème pixel est en ligne 3 et colonne 2.

(c) Le 1er pixel est en ligne `li - 1` et colonne `co - 1`.
Le 2ème pixel est en ligne `li - 1` et colonne `co + 1`.

2. (a) Pour que `image[li][co]` prenne la valeur 1, il faut que la condition suivante soit vérifiée : `image[li-1][co-1] != image[li-1][co+1]`

(b) On doit remplir le code de la façon suivante :

```
def remplir_ligne(image, li):
    image[li][0] = 0
    image[li][7] = 0
    for co in range(1, 7):
        if image[li-1][co-1] != image[li-1][co+1] :
            image[li][co] = 1
```

(c) On peut écrire le code suivant :

```
def remplir(image) :
    for li in range(1, len(image)):
        remplir_ligne(image, li)
```

Partie B

1. (a) On a :

$$\underline{00101100}_2 = 2^5 + 2^3 + 2^2 = 32 + 8 + 2 = 42$$

(b) On peut écrire le code suivant :

```
def conversion2_10(tab) :
    res = 0
    for k in range(8) :
        res = res + 2**(7 - k)
    return res
```

(c) On a :

$$78 = 64 + 8 + 4 + 2 = 2^6 + 2^3 + 2^2 + 2^1 = \underline{01001110}_2$$

Remarque : On peut aussi utiliser l'algorithme des divisions euclidiennes par 2 successives.

2. (a) Une ligne commence par une case blanche et finit par une case blanche donc l'écriture du nombre n en binaire commence par un `'0'` et finit par un `'0'`. Il s'agit donc d'un nombre pair et inférieur à 127.

(b) On peut compléter le code de la façon suivante :

```
def generer(n, k):
    tab = [None for i in range(k)]
    image = [[0 for j in range(8)] for i in range(k+1)]
    image[0] = conversion10_2(n)
    remplir(image)
    for i in range(k):
        tab[i] = conversion2_10(image[i+1])
    return tab
```