Université de Bordeaux **Licence S&TS Mention Informatique, S1**

4TPU146U **2022/2023**

Informatique 1
TP noté du lundi 16 Janvier 2023
Durée : 1h30.

Exercice 1.

1. Écrire une fonction `estMultipleDe(n:int, d:int) -> bool` qui retourne `True` si l'entier `n` est un multiple de l'entier `d` et `False` sinon.
2. En utilisant, la fonction `estMultipleDe`, écrire une fonction `estPair(n:int) -> bool` qui retourne `True` si l'entier `n` est pair et `False` sinon.
3. Écrire une fonction `nbPairs(l:list) -> int` qui retourne le nombre d'entiers pairs dans la liste `l`.
4. Écrire une fonction `listePairs(l:list) -> list` qui retourne la liste des entiers pairs de `l`.

Exemples :

```
>>> estMultipleDe(13, 4)
False
>>> estMultipleDe(12, 4)
True
>>> estPair(6)
True
>>> estPair(3)
False
>>> existePair([7,1,3,5])
False
>>> existePair([7,1,2,3,5])
True
>>> l
[1, 3, 4, 3, 6, 7, 10, 2]
>>> nbPairs(l)
4
>>> listePairs(l)
[4, 6, 10, 2]
```

Exercice 2.
Une suite de Syracuse $S(a) = (u_n)_{n\in\mathbb{N}}$ est définie par récurrence de la façon suivante :

$$\begin{cases} u_0 & = a \\ u_{n+1} & = \begin{cases} u_n/2 & \text{si } u_n \text{ est pair,} \\ 3u_n + 1 & \text{sinon.} \end{cases} \end{cases}$$

où a est un entier naturel strictement positif.

5. Donner les 6 premiers termes de la suite $S(6)$.
6. Écrire une fonction `suivant(un:int) -> int` qui calcule u_{n+1} en fonction de u_n.
7. Écrire une fonction `suite(n:int, a:int) -> int` qui retourne le terme u_n de la suite $S(a)$.
8. Écrire une fonction `rangPourUn(a:int) -> int` qui retourne le plus petit entier n tel que $u_n = 1$ dans la suite $S(a)$.

```
>>> suivant(4)
2
>>> suivant(3)
10
>>> for n in range(5):
        print(n, ":␣", suite(n, 4))
0 :  4
1 :  2
2 :  1
3 :  4
4 :  2
>>> rangPourUn(1)
0
>>> rangPourUn(4)
2
>>> rangPourUn(6)
8
>>> rangPourUn(100)
25
```

Exercice 3.
Le but de l'exercice est de construire une image contenant le drapeau tchèque présenté dans la figure 1.

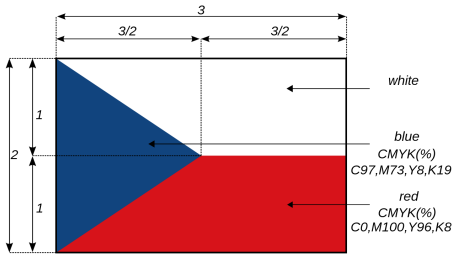


FIGURE 1 – Drapeau tchèque

9. Écrire une fonction `segmentHorizontal(img:image, x1:int, x2:int, y:int, c:couleur)`, qui trace une ligne horizontale allant de `x1` à `x2-1` à la hauteur `y`.
10. Écrire une fonction
    `rectanglePlein(img:image, x1:int, y1:int, x2:int, y2:int, c:couleur)`,
    qui dessine un rectangle plein dont le coin supérieur gauche a pour coordonnées (`x1, y1`) et le coin inférieur droit a pour coordonnées (`x2-1`, `y2-1`).

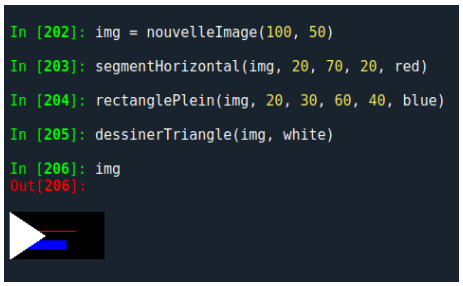


FIGURE 2 – Exemples

Soit la fonction booléenne `dansTriangle` définie ci-dessous, qui indique si le point de coordonnées (`x`, `y`) est situé dans le triangle correpondant à la zone bleue triangulaire du drapeau tchèque de hauteur `hauteur`.

```
def dansTriangle (x:int, y:int, h:int) -> bool:
    r = 2 / 3 * x
    return r - y <=0 and r + y - h <= 0
```

Exemples :

```
>>> dansTriangle(10, 10, 200)
True
>>> dansTriangle(100, 10, 200)
False
```

11. Écrire la fonction `trianglePlein(img:image, c:couleur)` qui colorie la zone triangulaire de `img` avec la couleur `c`.

12. Écrire une fonction `drapeauTcheque() -> image` qui retourne une nouvelle image de dimension $(300, 200)$ représentant le drapeau tchèque en respectant les proportions indiquées sur la figure 1.

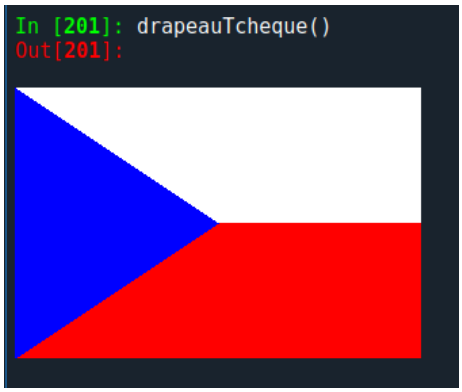


FIGURE 3 – Exemples

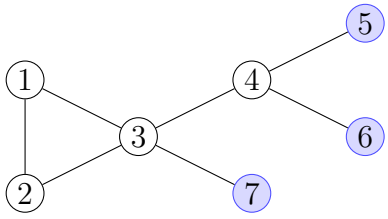
Exercice 4.

Dans un graphe sans sommet isolé, on appelle

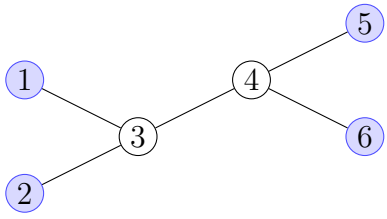
— *feuille* : un sommet de degré 1 et

— *sommet interne* : un sommet de degré strictement supérieur à 1.

Par exemple, le graphe de la figure 4a contient 3 feuilles et 4 sommets internes et celui de la figure 4b 4 feuilles et 2 sommets internes.



(a) Graphe à 3 feuilles



(b) Graphe à 4 feuilles

FIGURE 4

13. Écrire une fonction `estFeuille(s:sommet) -> bool` qui retourne `True` si le sommet `s` est une feuille et `False` sinon.
14. Écrire une fonction `estInterne(s:sommet) -> bool` qui retourne `True` si le sommet `s` est un sommet interne et `False` sinon.
15. Écrire une fonction `nbFeuillesAdjacentes(s:sommet) -> int` qui retourne le nombre de feuilles adjacentes au sommet `s`.
16. Écrire une fonction `toutSommetinterneAFeuilleAdjacente(g:graphe) -> bool` qui retourne `True` si **tous** les sommets internes du graphe ont **au moins une feuille adjacente** et `False` sinon. [1]

1. Le graphe de la figure 4a ne vérifie pas la propriété puisque les sommets 1 et 2 n'ont pas de feuille adjacente. Le graphe de la figure 4b vérifie la propriété.