1 En autonomie

Les questions suivantes sont le minimum à savoir après le cours, à réaliser en autonomie.

1. qu'est-ce qu'un algorithme récursif ?
2. qu'est-ce qu'un variant ?
3. On donne la fonction python suivante :

```python
def mystere(n: int) -> str:
    if n < 2:
        return str(n)
    return mystere(n // 2) + str(n % 2)
```

- écrire la trace de `mystere(11)`
- la fonction `mystere` est-elle récursive terminale ?
- donner un variant pour `mystere`

2 Parité

2.1 Pair-Impair

Voici la déclaration de deux fonctions :

```python
def est_pair(n: int) -> bool:
   return n == 0 or not est_impair(n)

def est_impair(n: int) -> bool:
   return not (n == 0 or est_pair(n))
```

1. Que pensez-vous des expressions calculées par chacune des deux fonctions dans les deux cas de base et récursif ?

2.2 Juste Pair

Voici la déclaration d'une fonction `est_pair`

```python
def est_pair(n: int) -> bool:
    if n == 0:
       res = True
    else:
       res = est_pair(n - 2)
    return res
```

1. Que pensez-vous de cette fonction ?

3 Somme de deux entiers

1. Proposez un algorithme récursif de calcul de la somme de deux entiers naturels a et b en supposant que les seules opérations de base dont vous disposez sont

- l'ajout de 1 à un entier x : $x + 1$
- le retrait de 1 à un entier x : $x - 1$
- et les comparaisons à 0 d'un entier x : $x = 0$, $x > 0$ et $x < 0$.

2. L'algorithme proposé est-il récursif terminal ?
3. Donnez un variant de l'algorithme proposé.
4. Programmez cet algorithme en python pour en faire une fonction `add` à deux paramètres.
5. Tracez les appels récursifs de `add(2, 3)`
6. Comment étendre cette fonction aux entiers de signe quelconque ?

4 Produit de deux entiers

1. Proposez un algorithme récursif de calcul du produit de deux entiers naturels a et b en supposant que les seules opérations de base dont vous disposez sont

- la somme de deux entiers x et y : $x + y$
- le retrait de 1 à un entier x : $x - 1$
- et la comparaison à 0 d'un entier x : $x = 0$.

2. L'algorithme proposé est-il récursif terminal ?
3. Donnez un variant possible de votre algorithme.
4. Programmez cet algorithme en python pour en faire une fonction `mult` à deux paramètres.
5. Tracez les appels récursifs de `mult(0, 3)` et de `mult(3, 2)`.

4.1 Puissance entière d'un nombre réel

1. Proposez un algorithme récursif de calcul de la puissance n-ième ($n \in \mathbb{N}$) d'un nombre réel a en supposant que les seules opérations de base dont vous disposez sont

- le produit de deux réels x et y : $x \times y$
- le retrait de 1 à un entier x : $x - 1$
- et la comparaison à 0 d'un entier x : $x = 0$

2. Exprimer le nombre de produit $c(n)$ effectués par votre algorithme en fonction de n.
3. (MI) En déduire un encadrement de $c(n)$ en fonction de la taille binaire t de n (*rappel* : $t(n) = \lfloor \log_2(n) \rfloor + 1$)
4. Nous allons améliorer la complexité de l'algorithme. En utilisant uniquement
   - une élévation au carré et
   - au plus une multiplication,

   exprimez a^n en fonction de a^k et a lorsque $n = 2k$, puis lorsque $n = 2k + 1$.
5. En déduire un autre algorithme récursif pour calculer a^n.

   Donnez, en fonction de n, un encadrement du nombre de multiplications effectuées par cet algorithme. Comparez avec l'algorithme précédent.

5 Nombre de chiffres d'un entier

Réalisez une fonction récursive qui calcule le nombre de chiffres de l'écriture décimale d'un nombre entier passé en paramètre *sans utiliser de chaîne de caractères.*

```
nbre_chiffres(0)
1
nbre_chiffres(2020)
4
```

6 Divisibilité par trois

Il est bien connu qu'un nombre entier est divisible par trois si et seulement si la somme des chiffres de son écriture décimale l'est aussi. Voilà donc un bel algorithme récursif.

1. Étudiez le (ou les) cas de base nécessaire(s). Qu'est ce qui garantit l'arrêt de cet algorithme ?
2. Écrivez une fonction récursive de calcul de la somme des chiffres de l'écriture décimale d'un entier.
3. Écrivez un prédicat récursif renvoyant `True` si, et seulement si, l'entier passé en paramètre est divisible par trois *sans utiliser l'opérateur* `%` ni la fonction `str`.
4. Donnez un variant de l'algorithme `est_divisible_par_trois` garantissant son arrêt

```
somme_chiffres(9876)
30
est_divisible_par_trois(9876)
True
somme_chiffres(9877)
31
est_divisible_par_trois(9877)
False
```

7 Algorithme d'Euclide

L'algorithme d'Euclide permet de calculer le pgcd de deux nombres entiers, c'est–à–dire le plus grand entier positif divisant ces deux nombres, par des divisions successives.

Voici le déroulement de cet algorithme pour le calcul du pgcd de $a = 119$ et $b = 544$

$$\begin{array}{rcrcrcr} 119 & = & 544 & \times & 0 & + & 119 \\ 544 & = & 119 & \times & 4 & + & 68 \\ 119 & = & 68 & \times & 1 & + & 51 \\ 68 & = & 51 & \times & 1 & + & \boxed{17} \\ 51 & = & \boxed{17} & \times & 3 & + & 0 \end{array}$$

Le pgcd de 119 et 544 est le dernier reste non nul, c'est–à–dire 17.

Le pgcd n'est pas défini lorsque les deux nombres sont nuls.

1. Exprimez de manière récursive cet algorithme. Vous pourrez supposer que les deux entiers a et b sont positifs ou nuls, et que l'un au moins de ces deux entiers n'est pas nul.
2. Codez cet algorithme en python en utilisant l'opérateur modulo.

8 Occurrences dans une chaîne

1. Réalisez une fonction récursive `nb_occurrences` d'entête

qui renvoie le nombre d'occurrences du caractère (second paramètre) dans la chaîne (premier paramètre).

Par exemple, `nb_occurrences('abcdebdb','z')` vaut 0 et `nb_occurrences('abcdebdb','b')` vaut 3.

2. Réalisez maintenant une fonction récursive `indice` paramétrée de la même façon que la fonction précédente
   - qui renvoie l'indice de la première occurrence du caractère dans la chaîne si le caractère est effectivement présent,
   - et qui déclenche une exception `ValueError: char not found`

   Cette fonction `indice` est en fait une méthode existante des chaînes de caractères nommée `index`. Une solution possible pour cette question est donc

```
def indice(s,c):
    return s.index (c)
```

mais ce n'est évidemment pas ce qui est attendu.

9 récursivité et liste

1. Écrire une fonction python récursive d'entête :

```
def liste_somme_rec(liste: list[int]) -> int:
    """Renvoie la somme des éléments de liste.
    précondition:     Exemples:
    $$$ liste_somme_rec([])
    0
    $$$ liste_somme_rec([1, 2, 3])
    6
    """
```

2. Écrire une fonction python récursive d'entête :

```
def liste_produit_rec(liste: list[int]) -> int:
    """Renvoie la somme des éléments de liste.
    précondition:     Exemples:
    $$$ liste_produit_rec([])
    1
    $$$ liste_produit_rec([2, 3, 5])
    30
    """
```

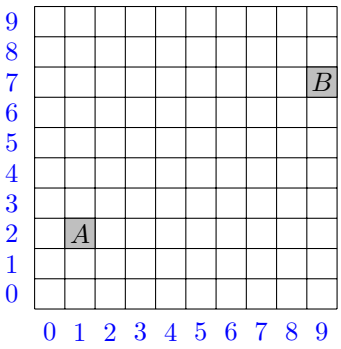
3. Écrire une fonction python récursive d'entête :

```
def split(liste: list[int]) -> tuple[list[int], list[int]]:
    """Renvoie un couple de (l1, l2) de deux listes.
    La liste l1 contient dans l'ordre exactement les éléments d'indice pair de liste.
    la liste l2 contient dans l'ordre exactement les éléments d'indice impair de liste.
    précondition:     Exemples :
    $$$ split([])
    ([], [])
    $$$ split([3, 1, 4, 1, 5, 9])
    ([3, 4, 5], [1, 1, 9])
    """
```

10 Tracer un segment

En informatique graphique, tous les points ont des coordonnées entières. Soit donc deux points A de coordonnées (x_A, y_A) et B de coordonnées (x_B, y_B).

Proposez un algorithme récursif pour tracer à l'écran le segment $[A, B]$. La commande primitive que vous utiliserez est la commande `plot(x, y)` qui dessine le pixel de coordonnées (x, y).



11 Coloriage

Supposons données les coordonnées (entières) (x, y) d'un pixel situé à l'intérieur d'une région du plan délimitée par une courbe fermée. Les points sur la courbe délimitant la région sont de couleur noire, et ceux à l'intérieur sont blancs. On souhaite donner la couleur rouge à tous les points à l'intérieur de la région.

Proposez un algorithme récursif pour effectuer ce coloriage. Vous pourrez utiliser deux fonctions `get_color(x, y)` qui renvoie la couleur du pixel de coordonnées (x, y) et `set_color(x, y, color)` qui fixe la couleur du pixel de coordonnées (x, y).

on supposera définie trois constantes BLANC, ROUGE et NOIR représentant les trois couleurs