❑ **Exercice 1** : *Exécution de programmes, recherche et corrections de bugs* 🎓 : *2022 Asie-Pacifique*

1. On considère la fonction `somme` prenant en paramètre un entier `n` strictement positif et qui renvoie le résultat du calcul $1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$:

```
def somme(n):
    total = 0
    for i in range(n):
        total = total + 1/i
    return total
```

Lors de l'exécution de `somme(10)`, le message d'erreur : `"ZeroDivisionError: division by zero"` apparaît. Identifier le problème et corriger la fonction pour qu'elle effectue le calcul demandé.

2. On considère la fonction `maxi` qui prend en argument une liste `liste` de nombres et renvoie le plus grand nombre de cette liste :

```
def maxi(liste):
    maximum = 0
    while  indice <= len(liste):
        if liste[indice] > maximum :
            maximum = liste[indice]
        indice = indice + 1
    return maximum
```

a) L'exécution de `maxi([2, 4, 9, 1])` déclenche une erreur, laquelle ? Corriger le programme afin que cette erreur ne se produise pas.

b) Le bug précédent est maintenant corrigé. Que renvoie à présent l'exécution de `maxi([-2, -7, -3])` ? Quel est le maximum de cette liste ? Modifier la fonction pour qu'elle renvoie le bon résultat

c) Ecrire cette fonction en utilisant une boucle `for` à la place de la boucle `while`.

3. On considère la fonction `suite` qui prend en argument un entier positif `n` et renvoie un entier :

```
def suite(n):
    if n==0:
        return 0
    else:
        return 3+2*suite(n-2)
```

a) Quelle valeur renvoie l'appel `suite(6)` ?

b) Que se passe-t-il si on exécute `suite(7)` ?

❑ **Exercice 2** : *programmation et récursivité* 🎓 : *2022 Polynésie*

On s'intéresse dans cet exercice à la construction de chaînes de caractères suivant certaines règles de construction.

Règle A : une chaîne est construite suivant la règle A dans les deux cas suivants :

- soit elle est égale à `"a"`
- soit elle est de la forme `"a" + chaine + "a"` où `chaine` est une chaine de caractères construite suivant la règle A.

Règle B : une chaîne est construite suivant la règle B dans les deux cas suivants :

- soit elle est de la forme `"b" + chaine + "b"` où `chaine` est une chaine de caractères construite suivant la règle A.
- soit elle est de la forme `"b" + chaine + "b"` où `chaine` est une chaine de caractères construite suivant la règle B.

On a reproduit ci-dessous l'aide de la fonction `choice` du module random.

```
>>>from random import choice
>>>help(choice)
Help on method choice in module random:
choice(seq) method of random.Random instance
Choose a random element from a non-empty sequence.
```

La fonction `A` ci dessous ne prend pas d'argument et renvoie une chaîne de caractères construite suivante la règle A, en choisissant aléatoirement entre les deux cas de figure de cette règle.

```
def A():
    if choice([True, False]):
        return "a"
    else:
        return "a" + A() + "a"
```

1. **a)** Cette fonction est-elle récursive ? Justifier.
   b) L'appel `choice([True,False])` peut renvoyer `False` un très grand nombre de fois consécutives. Expliquer pourquoi ce cas de figure amènerait à une erreur d'exécution.

Dans la suite, on considère une deuxième version de la fonction `A`. A présent, elle prend en paramètre un entier `n` tel que si la valeur de `n` est négative ou nulle, la fonction renvoie `"a"`. Sinon, elle renvoie une renvoie une chaine de caractères construite suivant la règle A avec un `n` diminué de 1, en choisissant entre les deux cas de figure de cette règle.

```
def A(n):
    if .... or choice([True,False]):
        return "a"
    else:
        return "a" + ... + "a"
```

2. **a)** Recopier sur la copie et compléter aux emplacments des points de suspension `...` le code de cette nouvelle fonction `A`.
   b) Justifier le fait qu'un appel de la forme `A(n)` avec `n` un nombre entier positif inférieur à 50, termine toujours.

On donne ci-après le code de la fonction récursive `B` qui prend en paramètre un entier `n` et qui renvoie une chaine de caractères construite suivant le règle B.

```
def B(n):
    if n<=0 or choice([True,False]):
        return "b" + A(n-1) + "b"
    else:
        return "b" + B(n-1) + "b"
```

On admet que :

- Les appels `A(-1)` et `A(0)` renvoient la chaîne `"a"` ;
- l'appel `A(1)` renvoie la chaîne `"a"` ou la chaîne `"aaa"` ;
- l'appel `A(2)` renvoie la chaine `"a"` ou la chaîne `"aaa"` ou la chaîne `"aaaaa"`.

3. Donner toutes les chaînes possibles renvoyées par les appels `B(0)`, `B(1)` et `B(2)`.
4. **a)** Ecrire une fonction `raccourcir` qui prend comme paramètre une chaîne de caractères `chaine` de longueur supérieure ou égale à 2, et renvoie la chaîne de caractères obtenue en supprimant de `chaine` le premier et dernier caractère. On donne ci-dessous des exemples d'appel à cette fonction dans une console Python.

```
>>> raccourcir("python")
>>> "ytho"
>>> raccourcir("ab")
>>> ""
```

   b) Recopier sur la copie et compléter les points de suspension `...` du code de la fonction `regleA` ci-dessous pour qu'elle renvoie `True` si la chaine passée en paramètre est construite suivant la règle A et `False` sinon.

```
    def regleA(chaine):
        n = len(chaine)
    if n >= 2:
        return chaine[0] == "a" and chaine[n-1] == "a" and regleA(...)
    else:
        return chaine == ...
```