Terminale NSI Evaluation n°1 S1

Nom : Prénom : Note :

Exercice n° 1 -- 4 pts

On considère la fonction ci-contre.

```
def mystere(n1,n2):
    """
    :param n1: un entier naturel
    :param n2: un entier naturel
    """
    if n1 == 0 or n2 == 0:
        return 0
    return n1 + mystere(n1,n2-1)
```

1. Quel est le résultat renvoyé par `mystere(5,3)` ?
2. Décrire en français ce que fait cette fonction.
3. Ecrire le code d'une version itérative de cette fonction en utilisant une boucle.

   L'utilisation du signe de la multiplication n'est pas autorisé

   Ne pas oublier de commenter la fonction avec les docstring et de prévoir des tests avec doctest
4. Quel est résultat renvoyé par `mystere(2,-1)` Expliquer.

Exercice n° 2 -- 4 pts

Une association a remarqué que d'une année à l'autre : - elle gagne 4% de nouveaux adhérents;
- elle perd 50 adhérents

1. En admettant que le nombre d'adhérents de cette association était égal à 2000 le 1er janvier 2019, écrire en Python une fonction récursive nommée nombre(n) qui affiche le nombre théorique d'adhérents après n années avec $n \geqslant 1$ n > 0.

2. Dans ce même programme, afficher le nombre théorique d'adhérents pour les 20 prochaines années.

Exercice n° 3 -- 2 pts

Pour la fonction récursive suivante : $a(n) = \begin{cases} 1 & \textbf{si } n = 0 \\ 2 \times a(2n) & \textbf{si } n \textbf{ est impair} \\ 1 + a(n-1) & \textbf{si } n \textbf{ est pair} \end{cases}$

déterminer un exemple d'appel qui ne se termine pas et justifier pourquoi éventuellement avec la liste des appels récursifs.

Exercice n° 4 -- 5 pts

Voici écrit en pseudo-code la fonction qui permet de rendre la monnaie pour un distributeur de boissons.

```
fonction rendu_pieces(somme,S)
    ind = 0
    pieces_rendues = liste vide
    tant que somme est différent de 0 Faire
        tant que S[ind] <= somme Faire
            ajouter S[ind] à la fin de pieces_rendues
            somme = somme - S[ind]
        si somme est différent de 0 alors
            si ind = longueur(S) - 1 alors
                Retourner "Rendu impossible"
            sinon
                ind = ind + 1
    retourner pieces_rendues
```

On donne le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`

```
import doctest

def rendu_pieces_recursif(somme,S,ind=0,pieces_rendues=[]):
    if somme == 0:
        return pieces_rendues
    if S[ind] <= somme:
        return rendu_pieces_recursif(..... ,S,ind,pieces_rendues +
    if ind == .....:
        return "Rendu impossible"
    return rendu_pieces_recursif(somme,S,.....,pieces_rendues)

doctest.testmod()
```

Exemple d'utilisation :

```
>>> rendu_pieces_recursif(2021,(500,100,50,10,5,2,1))
[500, 500, 500, 500, 10, 10, 1]
```

1. Compléter le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`.
2. Commenter cette fonction en utilisant les `docstrings`.
3. Prévoir plusieurs jeux de tests en utilisant la bibliothèque `doctest`.

Exercice n° 5 -- 5 pts

Sur votre compte Replit traiter le projet nommé **Sac à dos**.

Terminale NSI Evaluation n°1 S2

Nom : Prénom : Note :

Exercice n° 1 -- 4 pts

On considère la fonction ci-contre.

1. Quel est le résultat renvoyé par `mystere(2,3)` ?
2. Décrire en français ce que fait cette fonction.
3.
4. Quel est le résultat renvoyé par `mystere(-2,1)` ? Expliquer.

   Ecrire le code d'une version itérative de cette fonction en utilisant une boucle.

   Ne pas oublier de commenter la fonction avec les docstring et de prévoir des tests avec doctest

```
def mystere(n1,n2):
    """
       :param n1: un entier naturel
       :param n2: un entier naturel
    """
    if n1 == 0:
        return n2
    return mystere(n1 - 1,n2 + 1)
```

Exercice n° 2 -- 4 pts

Ecrire en Python une fonction récursive `pgcd(a,b)` qui renvoie le plus grand diviseur commun de deux nombres a et b.

Pour cela, on utilisera le résultat mathématique suivant :

pgcd(a,b) = pgcd(b,r) avec a = bq + r

Exercice n° 3 -- 2 pts

Pour la fonction récursive suivante : $a(n) = \begin{cases} 1 & \textbf{si } n = 0 \\ 2 \times a(2n) & \textbf{si } n \textbf{ est impair} \\ 1 + a(n-1) & \textbf{si } n \textbf{ est pair} \end{cases}$

déterminer un exemple d'appel qui ne se termine pas et justifier pourquoi éventuellement avec la liste des appels récursifs.

Exercice n° 4 -- 5 pts

Voici écrit en pseudo-code la fonction qui permet de rendre la monnaie pour un distributeur de boissons.

```
fonction rendu_pieces(somme,S)
    ind = 0
    pieces_rendues = liste vide
    tant que somme est différent de 0 Faire
        tant que S[ind] <= somme Faire
            ajouter S[ind] à la fin de pieces_rendues
            somme = somme - S[ind]
        si somme est différent de 0 alors
            si ind = longueur(S) - 1 alors
                Retourner "Rendu impossible"
            sinon
                ind = ind + 1
    retourner pieces_rendues
```

On donne le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`

```
import doctest

def rendu_pieces_recursif(somme,S,ind=0,pieces_rendues=[]):
    if somme == 0:
        return pieces_rendues
    if S[ind] <= somme:
        return rendu_pieces_recursif(..... ,S,ind,pieces_rendues +
    if ind == .....:
        return "Rendu impossible"
    return rendu_pieces_recursif(somme,S,.....,pieces_rendues)

doctest.testmod()
```

Exemple d'utilisation :

```
>>> rendu_pieces_recursif(2021,(500,100,50,10,5,2,1))
[500, 500, 500, 500, 10, 10, 1]
```

1. Compléter le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`.
2. Commenter cette fonction en utilisant les `docstrings`.
3. Prévoir plusieurs jeux de tests en utilisant la bibliothèque `doctest`.

Exercice n° 5 -- 5 pts

Sur votre compte Replit traiter le projet nommé **Sac à dos**.

Terminale NSI Evaluation n°1 S3

Nom : Prénom : Note :

Exercice n° 1 -- 4 pts

On considère la fonction ci-contre.

```
def mystere(n1,n2):
    """
       :param n1: un entier naturel
       :param n2: un entier naturel
    """
    if n1 == 0:
        return n2
    return mystere(n1 - 1,n2 + 1)
```

1. Quel est le résultat renvoyé par `mystere(2,3)` ?
2. Décrire en français ce que fait cette fonction.
3.
4. Quel est le résultat renvoyé par `mystere(-2,1)` ? Expliquer.

   Ecrire le code d'une version itérative de cette fonction en utilisant une boucle.

   Ne pas oublier de commenter la fonction avec les docstring et de prévoir des tests avec doctest

Exercice n° 2 -- 4 pts

Une association a remarqué que d'une année à l'autre :

1. En admettant que le nombre d'adhérents de cette association était égal à 2000 le 1er janvier 2019, écrire en Python une fonction récursive nommée `nombre(n)` qui affiche le nombre théorique d'adhérents après n années avec $n \geq 1$.
2. Dans ce même programme, afficher le nombre théorique d'adhérents pour les 20 prochaines années.

Exercice n° 3 -- 2 pts

Pour la fonction récursive suivante : $a(n) = \begin{cases} 1 & \textbf{si } n = 0 \\ 2 \times a(2n) & \textbf{si } n \textbf{ est impair} \\ 1 + a(n-1) & \textbf{si } n \textbf{ est pair} \end{cases}$

déterminer un exemple d'appel qui ne se termine pas et justifier pourquoi éventuellement avec la liste des appels récursifs.

Exercice n° 4 -- 5 pts

Voici écrit en pseudo-code la fonction qui permet de rendre la monnaie pour un distributeur de boissons.

```
fonction rendu_pieces(somme,S)
    ind = 0
    pieces_rendues = liste vide
    tant que somme est différent de 0 Faire
        tant que S[ind] <= somme Faire
            ajouter S[ind] à la fin de pieces_rendues
            somme = somme - S[ind]
        si somme est différent de 0 alors
            si ind = longueur(S) - 1 alors
                Retourner "Rendu impossible"
            sinon
                ind = ind + 1
    retourner pieces_rendues
```

On donne le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`

```
import doctest

def rendu_pieces_recursif(somme,S,ind=0,pieces_rendues=[]):
    if somme == 0:
        return pieces_rendues
    if S[ind] <= somme:
        return rendu_pieces_recursif(..... ,S,ind,pieces_rendues +
    if ind == .....:
        return "Rendu impossible"
    return rendu_pieces_recursif(somme,S,.....,pieces_rendues)

doctest.testmod()
```

Exemple d'utilisation :

```
>>> rendu_pieces_recursif(2021,(500,100,50,10,5,2,1))
[500, 500, 500, 500, 10, 10, 1]
```

1. Compléter le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`.
2. Commenter cette fonction en utilisant les `docstrings`.
3. Prévoir plusieurs jeux de tests en utilisant la bibliothèque `doctest`.

Exercice n° 5 -- 5 pts

Sur votre compte Replit traiter le projet nommé **Sac à dos**.

Terminale NSI Evaluation n°1 S4

Nom : Prénom : Note :

Exercice n° 1 -- 4 pts

On considère la fonction ci-contre.

```
def mystere(n1,n2):
    """
    :param n1: un entier naturel
    :param n2: un entier naturel
    """
    if n1 == 0 or n2 == 0:
        return 0
    return n1 + mystere(n1,n2-1)
```

1. Quel est le résultat renvoyé par `mystere(5,3)` ?
2. Décrire en français ce que fait cette fonction.
3. Ecrire le code d'une version itérative de cette fonction en utilisant une boucle.

   L'utilisation du signe de la multiplication n'est pas autorisé

   Ne pas oublier de commenter la fonction avec les docstring et de prévoir des tests avec doctest
4. Quel est résultat renvoyé par `mystere(2,-1)` Expliquer.

Exercice n° 2 -- 4 pts

Ecrire en Python une fonction récursive `pgcd(a,b)` qui renvoie le plus grand diviseur commun de deux nombres a et b.

Pour cela, on utilisera le résultat mathématique suivant :

pgcd(a,b) = pgcd(b,r) avec a = bq + r

Exercice n° 3 -- 2 pts

Pour la fonction récursive suivante : $a(n) = \begin{cases} 1 & \textbf{si } n = 0 \\ 2 \times a(2n) & \textbf{si } n \textbf{ est impair} \\ 1 + a(n-1) & \textbf{si } n \textbf{ est pair} \end{cases}$

déterminer un exemple d'appel qui ne se termine pas et justifier pourquoi éventuellement avec la liste des appels récursifs.

Exercice n° 4 -- 5 pts

Voici écrit en pseudo-code la fonction qui permet de rendre la monnaie pour un distributeur de boissons.

```
fonction rendu_pieces(somme,S)
    ind = 0
    pieces_rendues = liste vide
    tant que somme est différent de 0 Faire
        tant que S[ind] <= somme Faire
            ajouter S[ind] à la fin de pieces_rendues
            somme = somme - S[ind]
        si somme est différent de 0 alors
            si ind = longueur(S) - 1 alors
                Retourner "Rendu impossible"
            sinon
                ind = ind + 1
    retourner pieces_rendues
```

On donne le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`

```
import doctest

def rendu_pieces_recursif(somme,S,ind=0,pieces_rendues=[]):
    if somme == 0:
        return pieces_rendues
    if S[ind] <= somme:
        return rendu_pieces_recursif(..... ,S,ind,pieces_rendues +
    if ind == .....:
        return "Rendu impossible"
    return rendu_pieces_recursif(somme,S,.....,pieces_rendues)

doctest.testmod()
```

Exemple d'utilisation :

```
>>> rendu_pieces_recursif(2021,(500,100,50,10,5,2,1))
[500, 500, 500, 500, 10, 10, 1]
```

1. Compléter le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`.
2. Commenter cette fonction en utilisant les `docstrings`.
3. Prévoir plusieurs jeux de tests en utilisant la bibliothèque `doctest`.

Exercice n° 5 -- 5 pts

Sur votre compte Replit traiter le projet nommé **Sac à dos**.

Terminale NSI Evaluation n°1 S5

Nom : Prénom : Note :

Exercice n° 1 -- 4 pts

On considère la fonction ci-contre.

1. Quel est le résultat renvoyé par `mystere(2,5)` ?
2. Décrire en français ce que fait cette fonction.
3. Ecrire le code d'une version itérative de cette fonction en utilisant une boucle.

   Ne pas oublier de commenter la fonction avec les docstring et de prévoir des tests avec doctest.

```
def mystere(x,n):
    """
    :param x: un nombre réel
    :param n: un entier naturel
    """
    if n == 0:
        return 1
    return x * mystere(x, n - 1)
```

Exercice n° 2 -- 4 pts

Ecrire en Python une fonction récursive `pgcd(a,b)` qui renvoie le plus grand diviseur commun de deux nombres a et b.

Pour cela, on utilisera le résultat mathématique suivant :

pgcd(a,b) = pgcd(b,r) avec a = bq + r

Exercice n° 3 -- 2 pts

Pour la fonction récursive suivante : $a(n) = \begin{cases} 1 & \textbf{si } n = 0 \\ 2 \times a(2n) & \textbf{si } n \textbf{ est impair} \\ 1 + a(n-1) & \textbf{si } n \textbf{ est pair} \end{cases}$

déterminer un exemple d'appel qui ne se termine pas et justifier pourquoi éventuellement avec la liste des appels récursifs.

Exercice n° 4 -- 5 pts

Voici écrit en pseudo-code la fonction qui permet de rendre la monnaie pour un distributeur de boissons.

```
fonction rendu_pieces(somme,S)
    ind = 0
    pieces_rendues = liste vide
    tant que somme est différent de 0 Faire
        tant que S[ind] <= somme Faire
            ajouter S[ind] à la fin de pieces_rendues
            somme = somme - S[ind]
        si somme est différent de 0 alors
            si ind = longueur(S) - 1 alors
                Retourner "Rendu impossible"
            sinon
                ind = ind + 1
    retourner pieces_rendues
```

On donne le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`

```
import doctest

def rendu_pieces_recursif(somme,S,ind=0,pieces_rendues=[]):
    if somme == 0:
        return pieces_rendues
    if S[ind] <= somme:
        return rendu_pieces_recursif(..... ,S,ind,pieces_rendues +
    if ind == .....:
        return "Rendu impossible"
    return rendu_pieces_recursif(somme,S,.....,pieces_rendues)

doctest.testmod()
```

Exemple d'utilisation :

```
>>> rendu_pieces_recursif(2021,(500,100,50,10,5,2,1))
[500, 500, 500, 500, 10, 10, 1]
```

1. Compléter le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`.
2. Commenter cette fonction en utilisant les `docstrings`.
3. Prévoir plusieurs jeux de tests en utilisant la bibliothèque `doctest`.

Exercice n° 5 -- 5 pts

Sur votre compte Replit traiter le projet nommé **Sac à dos**.

Terminale NSI Evaluation n°1 S6

Nom : Prénom : Note :

Exercice n° 1 -- 4 pts

On considère la fonction ci-contre.

```
def mystere(x,n):
    """
    :param x: un nombre réel
    :param n: un entier naturel
    """
    if n == 0:
        return 1
    return x * mystere(x, n - 1)
```

1. Quel est le résultat renvoyé par `mystere(2,5)` ?
2. Décrire en français ce que fait cette fonction.
3. Ecrire le code d'une version itérative de cette fonction en utilisant une boucle.

   Ne pas oublier de commenter la fonction avec les docstring et de prévoir des tests avec doctest.

Exercice n° 2 -- 4 pts

Une association a remarqué que d'une année à l'autre :

1. En admettant que le nombre d'adhérents de cette association était égal à 2000 le 1er janvier 2019, écrire en Python une fonction récursive nommée `nombre(n)` qui affiche le nombre théorique d'adhérents après n années avec $n \geq 1$.
2. Dans ce même programme, afficher le nombre théorique d'adhérents pour les 20 prochaines années.

Exercice n° 3 -- 2 pts

Pour la fonction récursive suivante : $a(n) = \begin{cases} 1 & \textbf{si } n = 0 \\ 2 \times a(2n) & \textbf{si } n \textbf{ est impair} \\ 1 + a(n-1) & \textbf{si } n \textbf{ est pair} \end{cases}$

déterminer un exemple d'appel qui ne se termine pas et justifier pourquoi éventuellement avec la liste des appels récursifs.

Exercice n° 4 -- 5 pts

Voici écrit en pseudo-code la fonction qui permet de rendre la monnaie pour un distributeur de boissons.

```
fonction rendu_pieces(somme,S)
    ind = 0
    pieces_rendues = liste vide
    tant que somme est différent de 0 Faire
        tant que S[ind] <= somme Faire
            ajouter S[ind] à la fin de pieces_rendues
            somme = somme - S[ind]
        si somme est différent de 0 alors
            si ind = longueur(S) - 1 alors
                Retourner "Rendu impossible"
            sinon
                ind = ind + 1
    retourner pieces_rendues
```

On donne le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`

```
import doctest

def rendu_pieces_recursif(somme,S,ind=0,pieces_rendues=[]):
    if somme == 0:
        return pieces_rendues
    if S[ind] <= somme:
        return rendu_pieces_recursif(..... ,S,ind,pieces_rendues +
    if ind == .....:
        return "Rendu impossible"
    return rendu_pieces_recursif(somme,S,.....,pieces_rendues)

doctest.testmod()
```

Exemple d'utilisation :

```
>>> rendu_pieces_recursif(2021,(500,100,50,10,5,2,1))
[500, 500, 500, 500, 10, 10, 1]
```

1. Compléter le code d'une version récursive de cette fonction nommée `rendu_pieces_recursif()`.
2. Commenter cette fonction en utilisant les `docstrings`.
3. Prévoir plusieurs jeux de tests en utilisant la bibliothèque `doctest`.

Exercice n° 5 -- 5 pts

Sur votre compte Replit traiter le projet nommé **Sac à dos**.