Spécialité NSI en terminale
Exercices 1

1 Exercice 1

Les seules opérations autorisées sur les nombres entiers sont soit ajouter 1, soit retrancher 1. La fonction `somme` renvoie la somme des entiers positifs `a` et `b`.

```
def somme(a, b):
    while b > 0:
        a = a + 1
        b = b - 1
    return a
```

Écrire une version récursive de cette fonction.

2 Exercice 2

En s'inspirant de l'exercice précédent, il est demandé d'écrire une fonction qui renvoie la somme de deux entiers quelconques. Les seules opérations autorisées sur les nombres entiers sont ajouter ou retrancher 1.

1. Écrire une fonction `somme` itérative avec une boucle `while`.
2. Écrire une fonction `somme_rec` récursive non terminale.
3. Écrire une fonction `somme_rec_term` récursive terminale.

Pour effectuer la somme $a + b$ dans les trois questions, distinguer les cas $b > 0$, $b < 0$ et $b = 0$.

3 Exercice 3

On reprend la fonction du cours permettant de savoir si un entier naturel est pair ou non.

```
def pair(n):
    while n > 0:
        n = n - 2
    return n == 0
```

Écrire une version récursive de cette fonction.

4 Exercice 4

1. Écrire une fonction récursive `puissance` qui prend en paramètres un flottant `x` non nul et un entier naturel `n` et qui renvoie $\texttt{x}^{\texttt{n}}$. On se base sur la définition mathématique : $x^0 = 1$ et $x^n = x \times x^{n-1}$ pour $n > 0$.
2. Dans la première question, on se contente de traduire la définition, et la version obtenue n'est pas récursive terminale. Modifier le code pour que la fonction soit récursive terminale. On peut s'inspirer des exemples du cours concernant la fonction factorielle.

5 Exercice 5

Il est demandé, comme à l'exercice précédent, d'écrire une fonction `puissance` qui prend en paramètres un nombre `x` et un entier naturel `n` et qui renvoie $\texttt{x}^{\texttt{n}}$. Mais la méthode est différente.

Pour cela, on note que $x^0 = 1$ et on remarque que si $n = 2k$, alors $x^n = x^{2k} = (x^k)^2$ et si $n = 2k+1$ alors $x^n = x^{2k+1} = x(x^k)^2$. Le problème est divisé en deux sous-problèmes.

6 Exercice 6

Il s'agit d'écrire une fonction `puissance`, une troisième version, qui prend en paramètres un nombre `x` et un entier naturel `n` et qui renvoie $\texttt{x}^{\texttt{n}}$.

Pour cela, on note que $x^0 = 1$ et on remarque que si $n = 2k$, alors $x^n = x^{2k} = (x^2)^k$ et si $n = 2k+1$ alors $x^n = x^{2k+1} = x(x^2)^k$.

Comme à l'exercice précédent, le problème est divisé en deux sous-problèmes.

Écrire une version récursive non terminale, une version récursive terminale et une version itérative.

7 Exercice 7

1. Écrire une fonction récursive `somme` qui prend en paramètre une liste de nombres et renvoie la somme des termes de cette liste.
2. Expliquer avec l'exemple `somme([4,7,2])` le déroulement de l'exécution de la fonction.
3. Que penser du coût en temps et en espace d'une telle fonction ?

8 Exercice 8

1. Expliquer quel est le résultat renvoyé par la fonction `mystere`.

```
def mystere(n):
    if n < 2:
        return str(n)
    else:
        return mystere(n//2) + str(n%2)
```

2. Écrire une fonction récursive `binaire` qui prend en paramètres un entier relatif `r` et un entier naturel `n` strictement positif, et qui renvoie la représentation en machine de `r` sur `n` bits. La méthode utilisée est celle du complément à deux et on n'écrira pas les 0 au début.
3. Compléter le code avec un compteur passé en paramètre afin d'obtenir un résultat composé de `n` caractères, donc avec éventuellement des 0 au début.