# TD 5 - Python, C, Haskell - Tail recursion, Siblings calls, Evaluation strategy

Hugo Bolloré[1]

[1]hugo.bollore@uvsq.fr

Dans ce TD nous utiliserons plusieurs langages, le C, le Haskell et le Python. Pour le langage Haskell, utilisez le site suivant : www.tutorialspoint.com
Pour le langage Python vous avez 2 choix:

1. Dans votre navigateur, allez sur le site https://www.online-python.com/
2. Installez le package *python3* sur votre installation linux, et utilisez la commande suivante pour lancer un script python :

   ```
   python myscript.py
   ```

## Exercice 5.1 Tail end

### Question 5.1.1 Tail recursion

Ecrivez en C une fonction *moyenne* qui calcule la moyenne d'un tableau d'entier en utilisant une récursion terminale.

### Question 5.1.2 Siblings calls

Copiez et compilez le code suivant avec le compilateur *gcc*. Utilisez les options suivantes pour générer deux versions différentes de votre programme :

- -O1
- -O1 -foptimize-sibling-calls

```c
#include <stdio.h>

int __attribute__ ((noinline)) sister(short a, short b)
{
  return a + b;
}

int brother(long long n)
{
  if (n == 1)
    return 1;
   else
     return sister(n%21, n%21);
}

int main()
{
  printf("%d\n", brother(44));
  return 0;
}
```

Utilisez ensuite la commande *objdump -d <nom_ du_ binaire>* pour observer le code machine généré. Cherchez le code correspondant à la fonction *brother* et comparez les deux versions, que remarquez vous (cherchez sur internet ce que fait l'instruction *call*) ? Expliquez pourquoi le compilateur à pu faire cette optimisation.

## Exercice 5.2 Evaluation strategy

### Question 5.2.1 Haskell

Exécutez le code suivant en Haskell qui permet d'afficher la taille d'une liste statique d'éléments. Expliquez le résultat obtenu et ce que vous pouvez en déduire concernant la stratégie d'évaluation utilisée dans le langage Haskell.

```haskell
main :: IO ()
main = print . length $ [ (122+15), (3*21), (50-4), (18/0) ]
```

### Question 5.2.2 Python

Écrivez un programme permettant de trouver si le langage Python utilise une méthode de passage des paramètres par valeur, par référence ou par partage. Pour tester le passage par partage un de vos appels de fonction devra utiliser le type list qui peut se déclarer statiquement comme dans l'exemple suivant et qui est mutable dans ce langage :

```python
def function_example(my_list):
    print(list)

list = [1,2,3,'soleil']
function_example(list)
print(list)
```