```
from IPython.display import HTML
HTML(url="https://deptinfo-ensip.univ-poitiers.fr/FILES/NB/files/nb2.css")
```

# 1 Arbres binaires

Les arbres binaires sont des cas particuliers de graphes. Il est conseillé de lire le début du notebook sur les graphes, avant ce qui suit, car certains élément peuvent être directement appliqués.

On s'efforcera ici de traiter quelques alorithmes particuliers sur les arbres binaires.

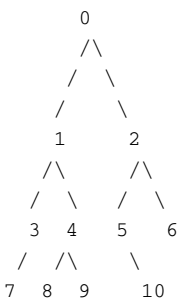
## 1.1 Représentation

On proposer de représenter un arbre binaire comme un triplet (sommet, fils gauche, fils droit) où fils gauche et fils droit seront eux-même des arbres binaires. Une feuille sera représentée par le triplet (sommet, None, None)

### 1.1.1 Création d'un arbre

Écrivez une fonction qui prend en paramètre un sommet, et deux arbres, et renvoie un arbre formé de ce sommet pour racine et des deux arbres passés en paramètre pour fils droit et fils gauche. Pour alléger l'écriture, on mettra des valeurs par défaut pour les fils gauche et fils droit (None).

Utilisez votre fonction pour créer l'arbre :




```
a1 = arbre(1, arbre(3, arbre(7), None), arbre(4, arbre(8), arbre(9)))
a2 = arbre(2, arbre(5, None, arbre(10)), arbre(6))
arb = arbre(0, a1, a2)
```

Notez qu'il est possible de créer l'ardre directement, en utilisant juste des tuples (la fonction arbre est juste là pour aider, et permet d'avoir un code plus évolutif) :

```
arb = (0, (1, (3, (7, None, None), None), (4, (8, None, None), (9, None, None))),
          (2, (5, None, (10, None, None)), (6, None, None))
      )
```

La partie affichage n'est pas nécessaire pour programmer les algorithmes qui suivent (un papier et un crayon suffisent). Néanmoins, si vous pensez avoir besoin de visualiser, réalisez d'abord la partie qui concerne l'affichage (fin du notebook).

## 1.2 Algorithmes sur les arbres

### 1.2.1 Mesures

- Nombre de noeuds d'un arbre
- Nombre de feuilles d'un arbre
- Hauteur d'un arbre

### 1.2.2 Parcours

- Parcours en largeur de l'arbre (voir aux files de priorité dequeue)
- Parcours en profondeur de l'arbre
- Parcours prefixe, infixe et postfixe

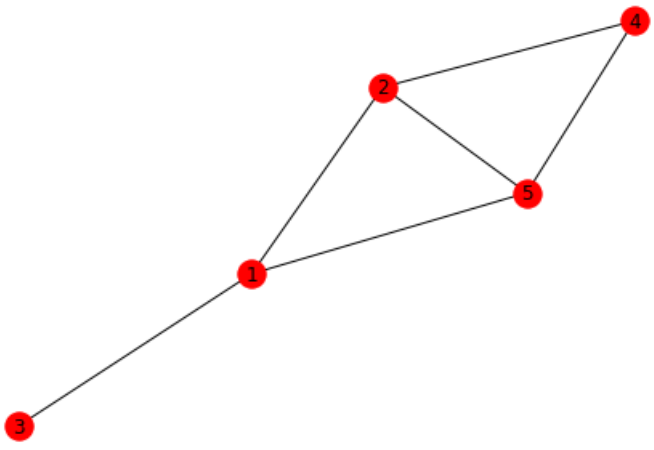
## 1.3 Affichage

En utilisant les bibliothèques des graphes (network), nous allons pouvoir tracer les arbres.

Écrivez une première fonction qui prend un arbre en paramètre et qui renvoie la liste des arêtes. Puis, utilisez cette liste d'arêtes pour constuire un graphe networkx et l'afficher.

Voici un exemple d'affichage d'un graphe avec networkx :

```
import networkx as nx
import matplotlib.pyplot as plt
%matplotlib inline
G = nx.Graph()
G.add_edges_from([(1, 2), (3, 1), (2, 4), (2, 5), (1, 5), (4, 5)])
nx.draw(G, with_labels=True)
plt.show()
```




png

Veillez à pouvoir tracer en écrivant simplement (l'appel à la fonction qui renvoie les arêtes de l'arbre sera **dans** la fonction `plot` que vous écrivez :

```
plot(arb)
```