

A) Matrices : recherche ... (2 points)

La fonction **searchMatrix(M, x)** retourne la position (*i, j*) de la première valeur **x** trouvée dans la matrice **M** non vide ou (-1, -1) si **x** n'existe pas dans **M**.

```
def searchMatrix (M, x):  
  
    (i, lin , col) = (0, len(M), len(M [0]) )  
    found = -1  
  
    while i < lin and found == -1:  
        j = 0  
        while j < col and M[i][j] != x:  
            j += 1  
        if j != col:  
            found = j  
        i += 1  
  
    if found != -1:  
        return (i -1, found )  
    else :  
        return (-1, -1)
```

La même fonction avec le return dans la boucle.

```
def searchMatrix2 (M, x):  
  
    (lin , col ) = (len(M), len(M [0]) )  
  
    for i in range (lin ):  
        for j in range (col ):  
            if M[i][j] == x:  
                return (i, j)  
  
    return (-1, -1)
```

B) Arbres binaires : branche de valeur maximum ... (6 points)

La fonction **maxbranch(B)** qui détermine la plus grande valeur des branches de l'arbre binaire **B** (dont les clés sont des entiers) est la suivante :

```
def maxbranch (B):  
    if B == None :  
        return 0  
    else :  
        return B.key + max( maxbranch (B. left ), maxbranch (B. right ))
```

C) Qu'est-ce que c'est ? ... (4 points)

1) Pour les applications suivantes de What :

- a) `what([(0,0), (10,10), (20, 20), (30, 30)], 15)`
- b) `what([(0,0), (10,20), (20,40), (30,60)], 24)`
- c) `what([(0,0), (1, 10), (2,100), (3, 1000)], 2.5)`
- d) `what([(0,3), (1,6), (2,9), (3,10), (4,15)], 20)`

Les résultats sont :

- a) 15
- b) 48
- c) 145
- d) 95

2) Si on considère les couples comme des coordonnées triées par ordre croissant, **what(L,Y)** calcule l'abscisse **X** correspondant à l'ordonnée **Y** calculée par interpolation linéaire.

D) Arbres binaires: symétrique ... (8 points)

La fonction **symmetric(B)** qui vérifie si un arbre binaire **B** est symétrique est :

```
def __symmetric(B1 , B2):
    if B1 == None or B2 == None :
        return B1 == B2
    else :
        if B1.key != B2.key:
            return False
        else :
            return __symmetric(B1.left , B2. right ) \
                and __symmetric(B1.right , B2. left )

def symmetric(B):
    return B == None or __symmetric(B.left , B. right )
```