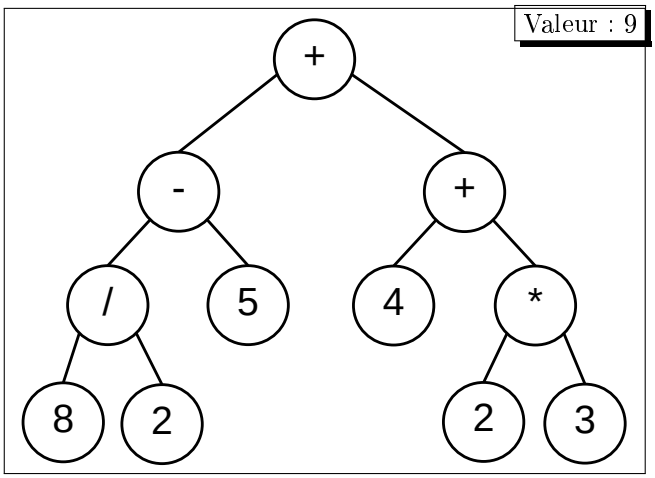
Algorithmique
Correction Contrôle n° 2 (C2)

Info-sup S2 – Epita

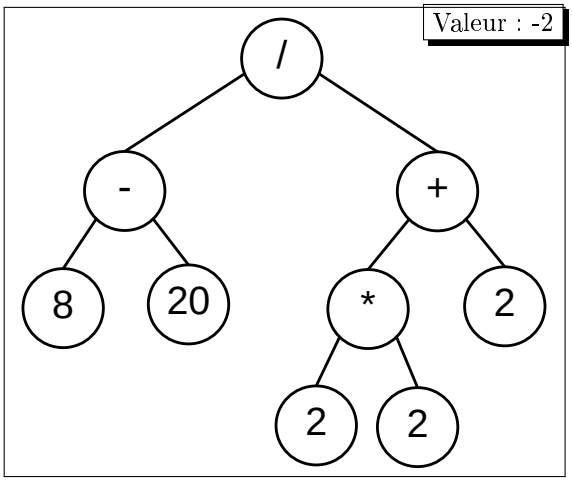
5 mars 2018 - 8 : 30

Solution 1 **(Dessine moi –** ***4 points*****)**

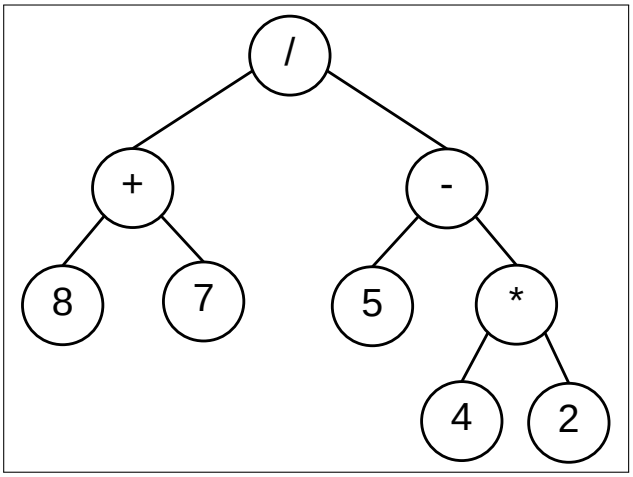
L'arbre B_1 :

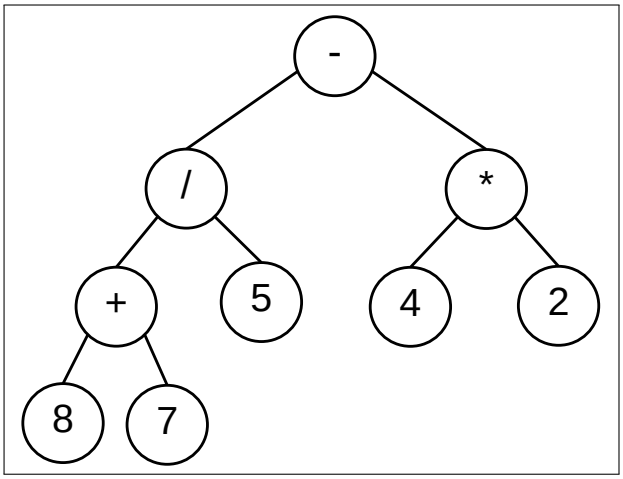


L'arbre B_2 :



L'arbre B_3 :





***Solution 2* (Compte moi – *3 points*)**

Spécifications :

La fonction `nodes(`B`)` calcule le nombre d'opérateurs *op* et le nombre d'opérandes *val* de l'arbre B et retourne le couple (*op*, *val*).

```
    def nbLeaves(B):
        if B.left == None:
            return 1
        else:
            return nbLeaves(B.left) + nbLeaves(B.right)

    def nodes(B):
        if B == None:
            return (0, 0)
        else:
            n = nbLeaves(B)
            return (n-1, n)

# ------------------------------------------------------------------------
    def nodes_rec(B):
        if B.left == None:
            return (0, 1)
        else:
            (int_left, ext_left) = nodes_rec(B.left)
            (int_right, ext_right) = nodes_rec(B.right)
            return (int_left + int_right + 1, ext_left + ext_right)

    def nodes2(B):   # useless here as B is not None !
        if B == None:
            return (0, 0)
        else:
            return nodes_rec(B)
```

***Solution 3* (Affiche moi – *2 points*)**

Spécifications :

La fonction `exp2str(`B`)` retourne une chaîne contenant l'expression, complètement parenthésée, représentée par l'arbre B.

```
def tree2expr(T):
    if T.left == None:
        return str(T.key)
    else:
        s = '('
        s = s + tree2expr(T.left)
        s = s + str(T.key)
        s = s + tree2expr(T.right)
        s  = s + ')'
        return s
# v2
def tree2expr2(T):
    if T.left == None:
       return str(T.key)
    else:
        return '(' + tree2expr(T.left) + str(T.key) + tree2expr(T.right) + ')'

# call
def exp2str(T):   # useless here as B is not None !
    if T == None:
        return ""
    else:
        return tree2expr(T)
```

Solution 4 **(Matrices : Symétrique -** ***4 points*****)**

Spécifications :

La fonction `isSymmetric(`A`)` teste si la matrice carrée A non vide est symétrique.

```
        def issymmetric(A):
            (i, n) = (0, len(A))
            sym = True
            while i < n and sym:
                j = 0
                while j < i and A[i][j] == A[j][i]:
                    j += 1
                sym = j == i
                i += 1
            return sym
```

Solution 5 **(Minimax –** ***4 points*****)**

La fonction `minimax(`M`)` retourne **la valeur minimale** parmi les maximums de chaque ligne de la matrice d'entiers M.

```
    def maxList(L):
        ''' maximum of list L, not empty '''
        m = L[0]
        for i in range(1, len(L)):
            m = max(m, L[i])
        return m

    def minimax(M):
        m = maxList(M[0])
        for i in range(1, len(M)):
            m = min(m, maxList(M[i]))
        return m

#-------------------------------------------------------------

    def minimax2(M):
        (n, p) = (len(M), len(M[0]))
        mini = M[0][0]
        for j in range (1, p):
            mini = max(mini, M[0][j])
        for i in range(1, n):
            maxi = M[i][0]
            for j in range(p):
                maxi = max(maxi, M[i][j])
            mini = min(mini, maxi)
        return mini
```

Solution 6 **(Mystery –** ***4 points*****)**

1. Résultats :

`what(A)` retourne 6

Matrice `A` après application :

	0	1	2
0	2	5	5
1	4	14	5
2	7	7	6

`what(B)` retourne 11

Matrice `B` après application :

	0	1	2
0	1	5	10
1	3	6	14
2	8	8	11

2. Nombre d'additions effectuées par `what(M)` avec `M` non vide de taille $n \times n$: $n^2 - 1$