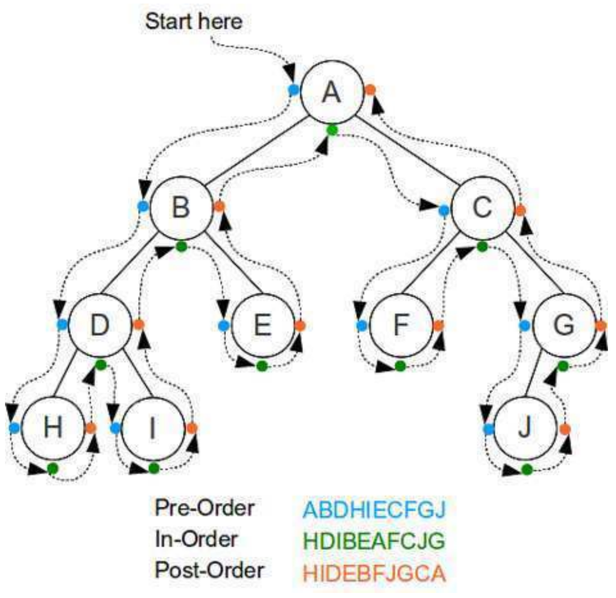
1 (Arbre Binaire : Construction - *2 points*)



2 (Arbre Binaire de Recherche - *4 points*)

```
PRECONDITIONS
  aucune
AXIOMES
  recherche(x,arbre-vide) = faux
  x = contenu(racine(B)) => recherche(x,B) = vrai
  x < contenu(racine(B)) => recherche(x,B) = recherche(x, g(B))
  x > contenu(racine(B)) => recherche(x,B) = recherche(x, d(B))

([88, 65, 64, 11, 59, 54, 13, 33, 51, 34, 46, 39, 40, 45, 44, 42], True)
([17, 89, 19, 57, 54, 26, 32, 36, 41, 46, 47, 93, 48, 60, 74, 88], False)
([94, 76, 74, 17, 63, 57, 52, 41, 39, 19, 35, 22, 31, 27, 26, 23], True)
([92, 32, 91, 36, 55, 56, 59, 79, 76, 73, 61, 10, 44, 11, 22, 31], False)
```

3 (Matrices : Symmétrique - *4 points*)

```python
def isSymmetric(A):
    l,c = len(A),len(A[0])
    if l != c:
        return False
    i,sym = 0,True
    while i < l and sym:
        j = 0
        while j < l and sym:
            sym = A[i][j] == A[j][i]
            j += 1
        i += 1
    return sym
```

4 (Arbre Binaire : Similarités - *5 points*)

```python
def postorder(B,l):
    if B != None:
        postorder(B.left, l)
        postorder(B.right, l)
        l.append(B.key)

def checkPostOrder(A, B):
    lA,lB = [],[]
    postorder(A,lA)
    postorder(B,lB)
    sA,sB = len(lA),len(lB)
    if sA != sB:
        return False
    i,c = 0,True
    while i < sA and c:
        c = sA[i] == sB[i]
        i += 1
    return c
```

5 (Arbre Binaire : PME - *6 points*)

```python
def pme(B):
    q = newQueue()
    enqueue(B, q)
    enqueue(None, q)
    (h, lce, nbe) = (0, 0, 0)
    while not isEmpty(q):
        B = dequeue(q)
        if B == None:
            h += 1
            if not isEmpty(q):
                enqueue(None, q)
        else:
            if B.left == B.right:
                lce += h
                nbe += 1
            else:
                if B.left != None:
                    enqueue(B.left, q)
                if B.right != None:
                    enqueue(B.right, q)
    return (lce/nbe)
```