In [1]:

```
class Node:
    def __init__(self, value, parent = None, left = None, right = None):
        self.value = value
        self.left = left
        self.right = right
        self.parent = parent

    def max_finder(self,height1, height2):
        if height1 > height2:
            return height1

        else:
            return height2

    def height_finder(self, given_root):
        if given_root == None:
            return -1

        else:
            return 1 + self.max_finder(self.height_finder(given_root.left),self.h

    def level_counter(self, root):
        if root.parent == None:
            return 1

        else:
            return 1 + self.level_counter(root.parent)

    def pre_order_traversal(self, the_root):
        if the_root != None:
            print(the_root.value,end=' ')
            self.pre_order_traversal(the_root.left)
            self.pre_order_traversal(the_root.right)

    def in_order_traversal(self, the_root):
        if the_root != None:
            self.in_order_traversal(the_root.left)
            print(the_root.value,end=' ')
            self.in_order_traversal(the_root.right)

    def post_order_traversal(self, the_root):
        if the_root != None:
            self.post_order_traversal(the_root.left)
            self.post_order_traversal(the_root.right)
            print(the_root.value,end=' ')

    def same_tree_checker(self,root,root2):
        if root == None and root2 == None:
            return 'These two trees are exactly same.'


        if root != None and root2 != None:
            return ((root.value == root2.value) and (self.same_tree_checker(root.
        else:
            return 'These two trees are not the same'
    def copy_a_tree(self,root):
        if root != None:
            copy_root = Node(root.value)
            copy_root.left = self.copy_a_tree(root.left)
            copy_root.right = self.copy_a_tree(root.right)

        else:
            return None

        return copy_root

'''
        1
       / \
      2   3
     / \   \
    4   5   6
'''
root = Node(1)
root_left1 = Node(2,root)
root_right1 = Node(3, root)
root.left = root_left1
root.right = root_right1

left1_left = Node(4, root_left1)
left1_right = Node(5, root_left1)

root_left1.left = left1_left
root_left1.right = left1_right

right1_right2  = Node(6,root_right1)
root_right1.right = right1_right2

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

root2 = Node(1)
root2_left1 = Node(2,root2)
root2_right1 = Node(3, root2)
root2.left = root2_left1
root2.right = root2_right1

dupli_left1_left = Node(4, root2_left1)
dupli_left1_right = Node(5, root2_left1)

root2_left1.left = left1_left
root2_left1.right = left1_right

dupli_right1_right2  = Node(6,root2_right1)
root2_right1.right = dupli_right1_right2

p1 = Node(root)

print('\n        ============= Task 1 =============     ')
print('The height of the tree is:',p1.height_finder(root))

print('\n        ============= Task 2 =============     ')
print('The level of the Node in the tree is:',p1.level_counter(left1_right))

print('\n        ============= Task 3 =============     ')
print('By using Pre-order traversal the elements will be: ',end='')
p1.pre_order_traversal(root)
print()

print('\n        ============= Task 4 =============     ')
print('By using In-order traversal the elements will be: ',end='')
p1.in_order_traversal(root)
print()

print('\n        ============= Task 5 =============     ')
print('By using Post-order traversal the elements will be: ',end='')
p1.post_order_traversal(root)
print()


print('\n        ============= Task 6 =============     ')

p2 = Node(None)
print(p2.same_tree_checker(root, root2))

print('\n        ============= Task 7 =============     ')

new_tree_root = p1.copy_a_tree(root)
p3 = Node(new_tree_root)
print('The copy of the given tree is (In pre order traversal):')
p3.pre_order_traversal(new_tree_root)

print('\n        ============= Task 8(a) =============     ')
```

```
        ============= Task 1 =============
The height of the tree is: 2

        ============= Task 2 =============
The level of the Node in the tree is: 3

        ============= Task 3 =============
By using Pre-order traversal the elements will be: 1 2 4 5 3 6

        ============= Task 4 =============
By using In-order traversal the elements will be: 4 2 5 1 3 6

        ============= Task 5 =============
By using Post-order traversal the elements will be: 4 5 2 6 3 1

        ============= Task 6 =============
These two trees are exactly same.

        ============= Task 7 =============
The copy of the given tree is (In pre order traversal):
1 2 4 5 3 6
        ============= Task 8(a) =============
```

The equivalent graph: