

CSE 220 - ASSIGNMENT 8

SADAF M. ANIS ID : 20101537 SEC : 08

In [19]:

```
1  #sada m.anis
2
3  class Node:
4      def __init__(self,value, left = None, right = None, parent = None):
5          self.value = value
6          self.left = left
7          self.right = right
8          self.parent = parent
9
10 class binary_tree:
11     def __init__(self,item):
12
13         self.item = Node(item)
14
15
16     #TASK 1
17     def height(self,root):
18         if root == None:
19             return -1
20
21         l_height = self.height (root.left)
22         r_height = self.height(root.right)
23
24         if l_height > r_height:
25             return l_height + 1
26         else:
27             return r_height + 1
28
29
30
31     #TASK 2
32
33     def level(self,root):
34         if root.parent == None:
35             return 1
36
37         return 1 + self.level(root.parent)
38
39     #TASK 3
40
41     def pre_order(self,new):
42
43         if new == None:
44             return
45
46         else:
47             print(new.value,end = " ")
48             self.pre_order(new.left)
49             self.pre_order(new.right)
50
51
52     #TASK 4
53
54     def in_order(self,new):
55         if new == None:
56             return
```

```
57
58         else:
59             self.in_order(new.left)
60             print(new.value, end = " ")
61             self.in_order(new.right)
62
63     #TASK 5
64
65     def pos_order(self,new):
66
67         if new == None:
68             return
69
70         else:
71             self.pos_order(new.left)
72             self.pos_order(new.right)
73             print(new.value, end = " ")
74
75
76
77     n1 = binary_tree(7)
78     n2 = binary_tree(10)
79     n3 = binary_tree(3)
80     n4 = binary_tree(5)
81     n5 = binary_tree(6)
82     n6 = binary_tree(30)
83     n7 = binary_tree(75)
84
85     root1 = n1
86
87
88     n1.item.left = n2.item
89
90     n2.item.parent = n1.item
91
92     n1.item.right = n3.item
93
94     n3.item.parent = n1.item
95
96     n2.item.left = n4.item
97
98     n4.item.parent = n2.item
99
100    n2.item.right = n5.item
101
102    n5.item.parent = n2.item
103
104    n3.item.right = n6.item
105
106    n6.item.parent = n3.item
107
108    n4.item.left = n7.item
109
110    n7.item.parent = n4.item
111
112
113
```

```

114 t1 = binary_tree(7)
115 t2 = binary_tree(10)
116 t3 = binary_tree(3)
117 t4 = binary_tree(5)
118 t5 = binary_tree(6)
119 t6 = binary_tree(30)
120 t7 = binary_tree(75)
121
122 root2 = t1
123
124 t1.item.left = t2.item
125
126 t2.item.parent = t1.item
127
128 t1.item.right = t3.item
129
130 t2.item.left = t4.item
131
132 t4.item.parent = t2.item
133
134 t2.item.right = t5.item
135 t5.item.parent = t2.item
136 t3.item.right = t6.item
137 t6.item.parent = t3.item
138 t4.item.left = t7.item
139 t7.item.parent = t4.item
140
141
142 print("TASK 1: ")
143 print("Height :", root1.height(root1.item))
144
145 print("TASK 2: ")
146 print("Level:", root1.level(root1.item.left.left.left))
147
148 print("TASK 3: ")
149 root1.pre_order(root1.item)
150
151 print("")
152 print("TASK 4: ")
153
154 root1.in_order(root1.item)
155 print("")
156 print("TASK 5: ")
157 root1.pos_order(root1.item)
158

```

```

TASK 1:
Height : 3
TASK 2:
Level: 4
TASK 3:
7 10 5 75 6 3 30
TASK 4:
75 5 10 6 7 3 30
TASK 5:
75 5 6 10 30 3 7

```

In [ ]:

1