Ques 1)

Ans → (c) In-order & level-order

↳ not possible

↳ due to both traversals doesn't provide info. regarding about the structure of the tree to uniquely construct it

(d) In-order & post-order

(e) post order & Pre order

Both have the same reason that not possible to uniquely identify a tree

Because Both traversal do not provide enough info. about the structure of the tree to uniquely construct it.

Date .....

In part (a) & part (b) it is possible to uniquely identify a tree for the given traversal combinations are → (a) postorder & level-order
(b) Inorder & pre-order.

Post order -& level order

(a) → possible because level-order traversal provides the order (in which Nodes are placed at each level)

post-order provides the order (in which nodes are visited after their child)

so, these two reasons Allowing us to recons. the tree uniquely.

Inorder -& pre order

(b) → possible because Inorder traversal provides the order in which nodes are visited [left subtree, root, right subtree]

and pre-order provides the root followed by left & right subtree allowing us to reconstruct the tree uniquely

Date .....

Pseudocode to construct tree

rt (a)

```
function cons_tree [postorder, level order]:
    if postorder is empty or levelorder is
                empty:
        return None

    root value = last element of post order
    root = new TreeNode (root Value)

    index = index of root value in level order

    leftlevelorder = levelorder [0:index]
    rightlevel.order = level.order [index+1:]

leftPostorder = elements of post order that
                are in leftlevel.order
rightPost order = elements of post order
                that are in rightlevelorde

root.left = cons_tree (leftPostorder,
                leftLevelorder)

root.right = cons-tree (rightPostorder,
                rightslevel order)

return root
```

Spiral

Part (b) →

```
function cons-tree(inorder, preorder):
    if inorder is empty or preorder is empty:
        return None

    rootValue = first element of preorder
    root = new TreeNode(rootValue)

    index = index of rootvalue in inorder

    leftinorder = inorder [0:index]
    rightinorder = inorder [index+1:]

    leftpreorder = elements of preorder that a
                   in leftinorder

    rightpreorder = elements of preorder that
                    are in rightinorder.

    root.left = cons-tree(leftinorder, left P

    root.right = cons-tree(rightinorder, rightP

    return root
```