1)

```
public void reverseQ(Queue<E> in){
        if (!in.isEmpty()){
                item = in.remove();
                reverseQ(in);
                in.add(item);
        }
}
```

2) c h d e g f b a

3)

```
boolean isValidBST ( BinaryT reeNode node , int min , int max ) {
        if ( node == null ) {
                return true ;
        }
        // Check if node key value is out of bounds
        if ( node . key < min || max < node . key ) {
                return false ;
        }
        return ( isValidBST ( node . leftChild , min , node . key -1) && isValidBST ( node .
rightChild , node . key +1 , max ));
}
```

4) Total number of internal nodes.

5) EC. The last element in post-order traversal is the root of the tree. Find it in the in-order expression. It must divide the in-order expression into two parts, which the left is the original left sub-tree elements and the right side vice versa. Counting from the back, you can find the same number of elements corresponded to the in-order expression in post-order expression. Then in those two parts in the post-order expression, find two new pivot elements again. Recursively do this procedure, you can uniquely decide the original tree.
However, with pre and post order, you cannot uniquely decide the original tree. For example: These two tree have same pre and post order expressions.