1. Remove the First Occurrence of Target from Doubly-linked List

solution

```
class Solution {
  public ListNode removeFirstOccurrence(ListNode head, int target) {
    ListNode prev = head.prev;
    ListNode curr = head;

    while(head != null){
      if (curr.val == target) {
        prev.next = curr.next;
        curr.next.prev = prev;
        curr.next = null;
        curr.prev = null;
      } else {
        curr = curr.next;
        pre = prev.next;
      }
    }
    return prev;
  }
}
```

2. Binary Tree In-order Traversal with Constant Space

solution

```
class Solution {
    public List<Integer> inorderTraversalWithConstantSpace(TreeNode root) {
        List<Integer> res = new ArrayList<>();
        Deque<TreeNode> deque = new ArrayDeque<>();
        pushLeftNodes(root, deque);

        while(!deque.isEmpty()){
          TreeNode curr = deque.removeFirst();
          res.add(curr.val);

          pushLeftNodes(curr.right, deque);
        }

        return res;
    }

    private void pushLeftNodes(TreeNode node, Deque<TreeNode> deque){
      while(node != null){
        deque.addFirst(node);
        node = node.left;
        }
      }
  }
```

3. Calculate Enclosed Territory

solution

```
class Solution {
  public int calculateEnclosed(char[][] territory) {

  }
}
```

4. The minimum path in Pyramid

solution

```

```