24. Swap Nodes in Pairs
We will go through the whole linked list and check whether there subsquent nodes next, if so, we will swap two nodes and until the end of the linked list. TC is O(n), SC is O(1)

```
class Solution:
    def swapPairs(self, head: ListNode) -> ListNode:
        dummy = ListNode(0)
        dummy_memo = dummy
        dummy.next = head
        while dummy.next and dummy.next.next:
            temp = dummy.next
            dummy.next = dummy.next.next
            temp.next = dummy.next.next
            dummy.next.next = temp
            dummy = dummy.next.next
        return dummy_memo.next
```

142. Linked List Cycle II

```
class Solution:
    def detectCycle(self, head: ListNode) -> ListNode:
        fast, slow = head, head
        if not fast or not fast.next:
            return None
        while fast and fast.next:
            fast = fast.next.next
            slow = slow.next
            if fast == slow:
                break
        if fast == slow and fast:
            start = head
            while start != fast:
                start = start.next
                fast = fast.next
            return fast
        else:
            return None
```

141. Linked List Cycle
We will use two pointers to detect whether there is a cycle in linked list. TC is O(n), SC is O(1)

```
class Solution:
    def hasCycle(self, head: ListNode) -> bool:
        fast, slow = head, head
        while fast and fast.next:
            fast = fast.next.next
            slow = slow.next
            if fast == slow:
                return True
        return False
```

206. Reverse Linked List
We will reverse the linked list using recursion. TC is O(n), SC is O(1)

```
class Solution:
    def reverseList(self, head: ListNode) -> ListNode:
        return self.reverse(head, None)

    def reverse(self, head, newList):
        if not head:
            return newList
        node = head.next
        head.next = newList
        return self.reverse(node, head)
```

218. The Skyline Problem
We will use heapq and the TC is O(nlogn)

```
from heapq import *
class Solution:
    def getSkyline(self, buildings: List[List[int]]) ->
List[List[int]]:
        events = [[L, -H, R] for L, R, H in buildings] + [[R, 0, 0]
for _, R, _ in buildings]
        events.sort()
        live = [[0, float('inf')]]
        result = [[0,0]]
        for x, negH, r in events:
            while live[0][1] <= x:
                heappop(live)
            if negH != 0:
                heappush(live, [negH, r])
            if live[0][0] +  result[-1][1] != 0:
                result.append([x, -live[0][0]])
        return result[1:]
```