**SET A Solution**

**def rearrange\_odd\_even(head):**

**if not head or not head.next:**

**return head # No need to rearrange if 0 or 1 node**

**odd\_head = odd\_tail = None**

**even\_head = even\_tail = None**

**current = head**

**# Separate odd and even nodes**

**while current:**

**next\_node = current.next # Store next node**

**current.next = current.prev = None # Detach node**

**if current.elem % 2 == 1: # Odd number**

**if not odd\_head:**

**odd\_head = odd\_tail = current**

**else:**

**odd\_tail.next = current**

**current.prev = odd\_tail**

**odd\_tail = current**

**else: # Even number**

**if not even\_head:**

**even\_head = even\_tail = current**

**else:**

**even\_tail.next = current**

**current.prev = even\_tail**

**even\_tail = current**

**current = next\_node # Move to next node**

**# If no odd nodes, return even list as is**

**if not odd\_head:**

**return even\_head**

**# Merge odd and even lists**

**odd\_tail.next = even\_head**

**if even\_head:**

**even\_head.prev = odd\_tail**

**return odd\_head # New head is the first odd node**

**Rubric (Set#A)**

| **SI** | **Category** | **Marks** |
| --- | --- | --- |
| 1 | Proper Method/Function Declaration | 1 |
| 2 | Initialise even and odd list | 1 |
| 3 | Proper iteration of List | 2 |
| 4 | Correct checking for even and odd node | 2 |
| 5 | Correctly adding nodes in the odd and even list | 4 |
| 6 | If no odd node present, return the even list | 2 |
| 7 | Merge odd and even lists | 2 |
| 8 | Return correct list | 1 |
| **Total = 15** | | |

**SET B Solution**

**def rearrange\_even\_odd(head):**

**if not head or not head.next:**

**return head # No need to rearrange if 0 or 1 node**

**even\_head = even\_tail = None**

**odd\_head = odd\_tail = None**

**current = head**

**# Separate even and odd nodes**

**while current:**

**next\_node = current.next # Store next node**

**current.next = current.prev = None # Detach node**

**if current.elem % 2 == 0: # Even number**

**if not even\_head:**

**even\_head = even\_tail = current**

**else:**

**even\_tail.next = current**

**current.prev = even\_tail**

**even\_tail = current**

**else: # Odd number**

**if not odd\_head:**

**odd\_head = odd\_tail = current**

**else:**

**odd\_tail.next = current**

**current.prev = odd\_tail**

**odd\_tail = current**

**current = next\_node # Move to next node**

**# If no even numbers, return the odd list as is**

**if not even\_head:**

**return odd\_head**

**# Merge even and odd lists**

**even\_tail.next = odd\_head**

**if odd\_head:**

**odd\_head.prev = even\_tail**

**return even\_head # New head is the first even node**

**Rubric (Set#B)**

| **SI** | **Category** | **Marks** |
| --- | --- | --- |
| 1 | Proper Method/Function Declaration | 1 |
| 2 | Initialise even and odd list | 1 |
| 3 | Proper iteration of List | 2 |
| 4 | Correct checking for even and odd node | 2 |
| 5 | Correctly adding nodes in the odd and even list | 4 |
| 6 | If no even node present, return the odd list | 2 |
| 7 | Merge odd and even lists | 2 |
| 8 | Return correct list | 1 |
| **Total = 15** | | |