**RUBRIC**

| **Category** | **Marks** |
| --- | --- |
| **Proper method/function definition with proper parameters and base case** | **3** |
| **Correct identification and summation of values at even positions** | **5** |
| **Correct identification and multiplication of values at odd positions** | **5** |
| **Correctly computing and returning the final result (sum − product)** | **2** |
| **Note: There can be multiple solutions which produce the correct answer. Use your best judgement to evaluate the scripts.** | |

**SET A:**

**Python:**

def sum\_even\_index(head, index):

if head is None:

return 0

if index % 2 == 0:

return head.data + sum\_even\_index(head.next, index + 1)

else:

return sum\_even\_index(head.next, index + 1)

def prod\_odd\_index(head, index):

if head is None:

return 1

if index % 2 == 1:

return head.data \* prod\_odd\_index(head.next, index + 1)

else:

return prod\_odd\_index(head.next, index + 1)

def sumProdAlternate(head):

return sum\_even\_index(head, 0) - prod\_odd\_index(head, 0)

**JAVA:**

public class labquiz4{

public static int sumEven(Node node, int index) {

if (node == null) {

return 0;

}

if (index % 2 == 0) {

return node.data + sumEven(node.next, index + 1);

} else {

return sumEven(node.next, index + 1);

}

}

public static int prodOdd(Node node, int index) {

if (node == null) {

return 1;

}

if (index % 2 == 1) {

return node.data \* prodOdd(node.next, index + 1);

} else {

return prodOdd(node.next, index + 1);

}

}

public static int sumProdAlternate(Node head) {

return sumEven(head, 0) - prodOdd(head, 0);

}

**SET B:**

**Python:**

def weightedSum(head):

return calcWeighted(head, 0)

def calcWeighted(node, index):

if node is None:

return 0

if index % 2 == 0:

return node.data \* 2 + calcWeighted(node.next, index + 1)

else:

return -node.data + calcWeighted(node.next, index + 1)

**JAVA:**

public class labquiz4{

public static int weightedSum(Node head) {

return calcWeighted(head, 0);

}

private static int calcWeighted(Node node, int index) {

if (node == null) return 0;

if (index % 2 == 0) {

return node.data \* 2 + calcWeighted(node.next, index + 1);

} else {

return -node.data + calcWeighted(node.next, index + 1);

}

}