

## Practical 9

**Aim : Write a Program to Swap Nodes pairwise**

**Algorithm :**

Algorithm for swap(Node head): (Data Swap)

1. Initialize:
  - Set l1 to the head of the linked list.
2. Iterate through the list:
  - While l1 is not null and l1.next is not null:
    - Store the data of l1 in a temporary variable temp.
    - Set l1.data to l1.next.data.
    - Set l1.next.data to temp.
    - Move l1 to l1.next.next (i.e., skip to the next pair).
3. Return:
  - Return the head of the modified linked list.

**Algorithm for addressSwap(Node head): (Node Swap)**

1. Initialize:
  - Create a dummy node with a value of -1 and set its next to head.
  - Set a pointer point to the dummy node.
2. Iterate through the list:
  - While point.next is not null and point.next.next is not null:
    - Set swap1 to point.next (the first node of the pair).
    - Set swap2 to point.next.next (the second node of the pair).

- Swap the nodes:
  - Set swap1.next to swap2.next (link the first node to the node after the second).
  - Set swap2.next to swap1 (link the second node to the first).
  - Set point.next to swap2 (link the previous node to the second node).
- Move point to swap1 (the first node of the newly swapped pair).

3. Return:

- Return dummy.next, which points to the new head of the modified linked list.

**Code:-**

```
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
    }
}
```

```
}
```

```
public class SwapNodesInPairs {
```

```
    // (data swap) if asked to swap data then  
    use this function
```

```
    public static Node swap(Node head) {
```

```
        Node l1 = head;
```

```
        while (l1 != null && l1.next != null) {
```

```
            int temp = l1.data;
```

```
            l1.data = l1.next.data;
```

```
            l1.next.data = temp;
```

```
            l1 = l1.next.next;
```

```
        }
```

```
        return head;
```

```
    }
```

```
    // (nodes swap) if asked to swap nodes then  
    use this function
```

```
public static Node addressSwap(Node head)
{
    Node l1 = head;
    Node dummy = new Node(-1);
    dummy.next = l1;
    Node point = dummy;

    while (point.next != null &&
point.next.next != null) {
        Node swap1 = point.next;
        Node swap2 = point.next.next;

        // Swapping the nodes
        swap1.next = swap2.next; // Link first
node to the node after the second
        swap2.next = swap1; // Link second
node to the first
    }
```

```
    point.next = swap2; // Link the
previous node to the second node
```

```
    // Move the pointer forward
    point = swap1; // Move to the next pair
}

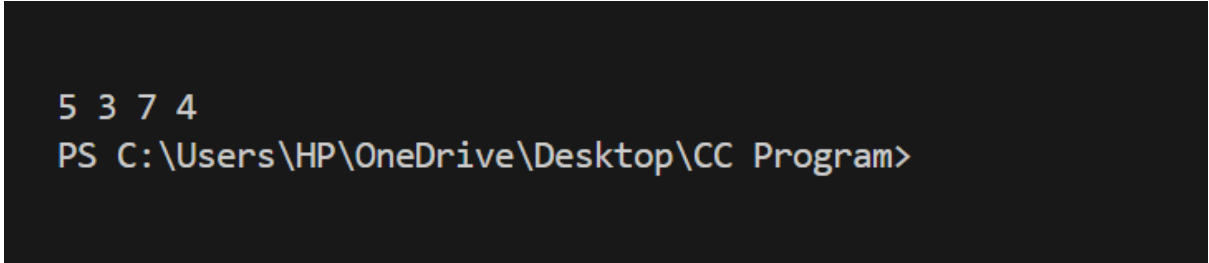
return dummy.next; // Return the new
head of the list
}
```

```
public static void main(String[] args) {
    Node l1 = new Node(3);
    l1.next = new Node(5);
    l1.next.next = new Node(4);
    l1.next.next.next = new Node(7);

    Node head = addressSwap(l1); // Swap
nodes in pairs
```

```
// Print the swapped list
while (head != null) {
    System.out.print(head.data + " ");
    head = head.next;
}
}
}
```

**Output:-**



```
5 3 7 4
PS C:\Users\HP\OneDrive\Desktop\CC Program>
```