Assignment 2:

1) Implement Singly LinkedList.

Source Code:

```
public class SinglyLinkedList {

    Node head;
    // Linked list Node.
    // This inner class is made static so that main() can access it
    static class Node {
        int data;
        Node next;

        Node(int d) {
            data = d;
            next = null;
        }
    }

    // Method to insert a new node
    public static SinglyLinkedList insert(SinglyLinkedList list, int data) {

        Node new_node = new Node(data);
        new_node.next = null;

        // If the Linked List is empty, then make the new node as head
        if (list.head == null) {
            list.head = new_node;
        } else {
            // Else traverse till the last node and insert the new_node there
            Node last = list.head;
            while (last.next != null) {
                last = last.next;
            }

            // Insert the new_node at last node
            last.next = new_node;
        }

        // Return the list by head
        return list;
    }

    // Method to print the SinglyLinkedList.
    public static void printList(SinglyLinkedList list) {
        Node currNode = list.head;

        System.out.print("SinglyLinkedList: ");

        // Traverse through the SinglyLinkedList
        while (currNode != null) {
            // Print the data at current node
```

```
            System.out.print(currNode.data + " ");

            // Go to next node
            currNode = currNode.next;
        }
    }

    public static SinglyLinkedList deleteAtPosition(SinglyLinkedList list, int index)
    {
        // Store head node
        Node currNode = list.head, prev = null;

        // CASE 1:If index is 0, then head node itself is to be deleted

        if (index == 0 && currNode != null) {
            list.head = currNode.next; // Changed head

            // Display the message
            System.out.println(
                index + " position element deleted");

            // Return the updated List
            return list;
        }

        // CASE 2: If the index is greater than 0 but less than the size of
SinglyLinkedList
        int counter = 0;
        while (currNode != null) {

            if (counter == index) {
                // Since the currNode is the required
                // position Unlink currNode from linked list
                prev.next = currNode.next;

                // Display the message
                System.out.println("\n"+ index + " Position element deleted");
                break;
            }
            else {
                // If current position is not the index continue to next node
                prev = currNode;
                currNode = currNode.next;
                counter++;
            }
        }

        // If the position element was found, it should be at currNode Therefore the
currNode shall not be null

        // CASE 3: The index is greater than the size of the SinglyLinkedList
        // In this case, the currNode should be null
        if (currNode == null) {
            // Display the message
```

```
            System.out.println(
                index + " position element not found");
        }
        return list;
    }
    public static void main(String[] args) {
        SinglyLinkedList list = new SinglyLinkedList();
        // Insert the values
        list = insert(list, 8);
        list = insert(list, 7);
        list = insert(list, 6);
        list = insert(list, 5);
        list = insert(list, 4);
        list = insert(list, 3);
        list = insert(list, 2);
        list = insert(list, 1);
        printList(list);
        deleteAtPosition(list, 3);
        printList(list);

    }
}
```

Output:

```
SinglyLinkedList: 8 7 6 5 4 3 2 1
3 Position element deleted
SinglyLinkedList: 8 7 6 4 3 2 1
```