Experiment 1.3

Student Name: Nishant Kumar mehta **UID: 21BCS3402**
Branch: CSE **Section/Group: IOT - 602**
Semester: 5th **Dateof Performance: 24/08/23**
Subject Name: Advance Programming Lab **Subject Code: 21CSP - 314**

1. Aim: Linked List: Demonstrate the concept of Linked List

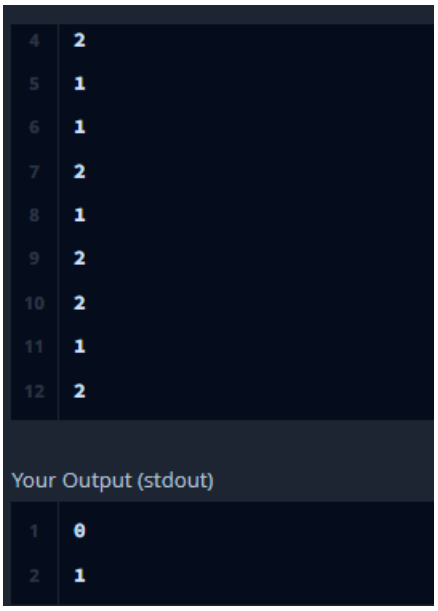
2. Objective:

- You're given the pointer to the head nodes of two linked lists. Compare the data in the nodes of the linked lists to check if they are equal. If all data attributes are equal and the lists are the same length, return 1. Otherwise, return 0s.
- Given a reference to the head of a doubly-linked list and an integer,data, create a new DoublyLinkedListNode object having data value and insert it at the proper location to maintain the sort.

3. Program and output:

```
static boolean compareLists(SinglyLinkedListNode head1, SinglyLinkedListNode head2)
{
    SinglyLinkedListNode temp1, temp2;
    temp1 = head1;
    temp2 = head2;
    while(temp1 != null && temp2 != null){
      if(temp1.data == temp2.data){
        temp1 = temp1.next;
        temp2 = temp2.next;
      } else {
        return false;
      }
```

```
        }
        if(temp1 == null && temp2 == null){
            return true;
        } else {
            return false;
        }
    }
```



2.

```
public static DoublyLinkedListNode sortedInsert(DoublyLinkedListNode llist, int data) {
  DoublyLinkedListNode current = llist;
    DoublyLinkedListNode lastNode = null;
    if (current == null) {
        current = new DoublyLinkedListNode(data);
        return current;
    }

    while (current != null) {
        if (data < current.data) {
            DoublyLinkedListNode tempNode = new DoublyLinkedListNode(data);
```

```
                tempNode.next = current;
                if (current.prev != null) {
                    tempNode.prev = current.prev;
                    tempNode.prev.next = tempNode;
                    current.prev = tempNode;
                } else {
                    current.prev = tempNode;
                    llist = tempNode;
                }
                break;
            }

            lastNode = current;
            current = current.next;
        }

        if (current == null) {
            lastNode.next = new DoublyLinkedListNode(data);
        }
        return llist;
    }
}
```