**Lab Task 10**



**Superior University Gold Campus**

|  |  |
| --- | --- |
| **Submitted to** | ****Mr. Rasikh Ali**** |
| **Submitted by** | **Javaid Ali** |
| **Roll No** | **SU92-BSSEM-S24-029 (Section – 3A)** |
| **Subject** | **Data Structures and Algorithms (Lab)** |
| **Class** | **BS – Software Engineering** |

# **Lab 10: Stack with Array and LinkedList**

**1-Stack With Array; Push, Pop, Display Functions**

**Functions:**

1. **push(d)** → Adds d at the top (end) of the stack.
2. **pop()** → Removes the top element.
3. **push\_at\_start(d)** → Inserts d at the **bottom** (start) of the stack.
4. **pop\_at\_start()** → Removes the **bottom** .
5. **display() →** Prints the stack from top to bottom.

**Why & How?**

* A **stack** normally follows **LIFO (Last-In-First-Out)**.
* push\_at\_start()**&**pop\_at\_start() shift elements to insert/delete at the bottom.
* The top variable tracks the current top index (-1 means empty).

**Outputs:**



**2- Stack With LinkedList; Push, Pop, Display Functions**

**Key Differences from Array-Based Stack:**

* Uses nodes (Stack objects) connected via pointers (next).
* No fixed size limit (unlike the array version).
* top is a global pointer tracking the top of the stack.

**Functions:**

1. **push(d)** → Adds d at the **top** (LIFO).
2. **pop()** → Removes the **top** element.
3. **push\_at\_start(d**) → Inserts d at the **bottom**.
4. **pop\_at\_start()** → Removes the **bottom** .
5. **display()** → Prints the stack from top to bottom.

**Why & How?**

* **Linked list** allows dynamic resizing (no overflow unless memory runs out).
* push\_at\_start()**&**pop\_at\_start() traverse the list to modify the **tail**.
* pop() simply moves top to top->next and deletes the old top.

**Outputs:**

