Stack Using Linked List

---

1. Node Structure

```
Node:
    val : integer
    next : pointer to Node
```

---

2. Check if Stack is Empty

Algorithm `is_empty(stack_top)` **Input:** Pointer to top node
Output: Boolean (true if empty, false otherwise)

```
1. If stack_top == NULL:
       Return true
2. Else:
       Return false
```

Time Complexity: O(1)

---

3. Create Node

Algorithm `create_node()` **Input:** None
Output: Pointer to new Node

```
1. Allocate memory for a new node, call it 'current'
2. If allocation fails:
       Print "Memory is full" and exit
3. Read value 'val' from user
4. Set current->val = val
5. Set current->next = NULL
6. Return current
```

Time Complexity: O(1)

---

4. Push / Insert

Algorithm `insert(stack_top)` **Input:** Pointer to top node
Output: Updated top node

```
1. temp = create_node()
2. temp->next = stack_top
3. stack_top = temp
4. Return stack_top
```

Time Complexity: O(1)

---

5. Display Stack (Iterative)

Algorithm `display(stack_top)` **Input:** Pointer to top node
Output: Print all elements

```
1. If stack_top == NULL:
       Print "Stack is empty" and return
2. temp = stack_top
3. While temp != NULL:
       Print temp->val
       temp = temp->next
```

Time Complexity: O(n)

---

6. Display Stack (Recursive)

Algorithm `rec_display(stack_top)`

```
1. If stack_top == NULL:
       Return
2. Print stack_top->val
3. Call rec_display(stack_top->next)
```

Time Complexity: O(n)

---

7. Pop / Delete Top Element

Algorithm `pop(stack_top)` **Input:** Pointer to top node
Output: Updated top node

```
1. If stack_top == NULL:
       Print "Stack is empty" and return stack_top
2. temp = stack_top
3. stack_top = stack_top->next
```

```
4. Free temp
5. Return stack_top
```

Time Complexity: O(1)

---

8. Peek Top Element

Algorithm `peek(stack_top)` **Input:** Pointer to top node
Output: Value at top or INT_MIN if empty

```
1. If stack_top == NULL:
       Print "Stack is empty"
       Return INT_MIN
2. Return stack_top->val
```

Time Complexity: O(1)

---

9. Main Menu Algorithm

```
1. Initialize top = NULL
2. Loop infinitely:
    a. Print menu:
       1) Insert
       2) Display
       3) POP
       4) PEEK
       5) Exit
    b. Read choice 'cho' from user
    c. Switch cho:
        Case 1: top = insert(top)
        Case 2: display(top)
        Case 3: top = pop(top)
        Case 4: Print peek(top)
        Case 5: Exit program
        Default: Print "Please enter correct input"
```

Time Complexity Summary

Operation	Complexity
insert	O(1)
pop	O(1)
peek	O(1)

Operation	Complexity
display	O(n)
rec_display	O(n)