Day 133/180 Introduction to Stack

1: Implement stack using array

```
// Definition of the MyStack class
class MyStack
{
private:
    int arr[1000]; // Array to store the elements of the stack
    int top;       // Variable to keep track of the top of the stack
public:
    // Constructor to initialize the stack
    MyStack() { top = -1; }
    // Function to push an element onto the stack
    void push(int x);
    // Function to pop an element from the top of the stack
    int pop();
};

// Implementation of the push function
void MyStack::push(int x) {
    // Increment the top pointer and insert the element at the top
    arr[++top] = x;
}

// Implementation of the pop function
int MyStack::pop() {
    // Check if the stack is empty
    if (top == -1)
        return -1; // Return -1 to indicate an empty stack

    // Retrieve the element from the top of the stack
    int ans = arr[top];

    // Decrement the top pointer to remove the element from the stack
    top--;

    // Return the popped element
    return ans;
}
```

2: Implement Stack using LinkedList

```
// Implementation of the push function
void MyStack::push(int x)
{
    // Create a new node with the given data
    StackNode* t = new StackNode(x);

    // Set the next pointer of the new node to the current top
    t->next = top;

    // Update the top pointer to the new node
    top = t;
}

// Implementation of the pop function
int MyStack::pop()
{
    // Initialize the answer to -1 (default for an empty stack)
    int ans = -1;

    // Check if the stack is not empty
    if (top != nullptr)
    {
        // Retrieve the data from the top node
        ans = top->data;

        // Move the top pointer to the next node (removing the top node)
        top = top->next;
    }

    // Return the popped element (or -1 for an empty stack)
    return ans;
}
```

3: Stack Operations

```
// Definition of the Geeks class
class Geeks
{
    // Function to insert element to stack
    public static void insert(Stack<Integer> st, int x)
    {
        st.push(x); // Using the push method to insert the element 'x' onto the
stack 'st'
    }

    // Function to pop element from stack
    public static void remove(Stack<Integer> st)
    {
        int x = st.pop(); // Using the pop method to remove the top element from
the stack 'st'
    }

    // Function to return the top of the stack
    public static void headOf_Stack(Stack<Integer> st)
    {
        int x = st.peek(); // Using the peek method to retrieve the top element
from the stack 'st'
        System.out.println(x + " "); // Printing the top element
    }

    // Function to find the element in the stack
    public static void find(Stack<Integer> st, int val)
    {
        // Using the contains method to check if the stack 'st' contains the value
'val'
        if(st.contains(val)){
            System.out.println("Yes"); // Print "Yes" if the element is found in
the stack
        }
        else{
            System.out.println("No"); // Print "No" if the element is not found in
the stack
```

```
        }
    }
}
```