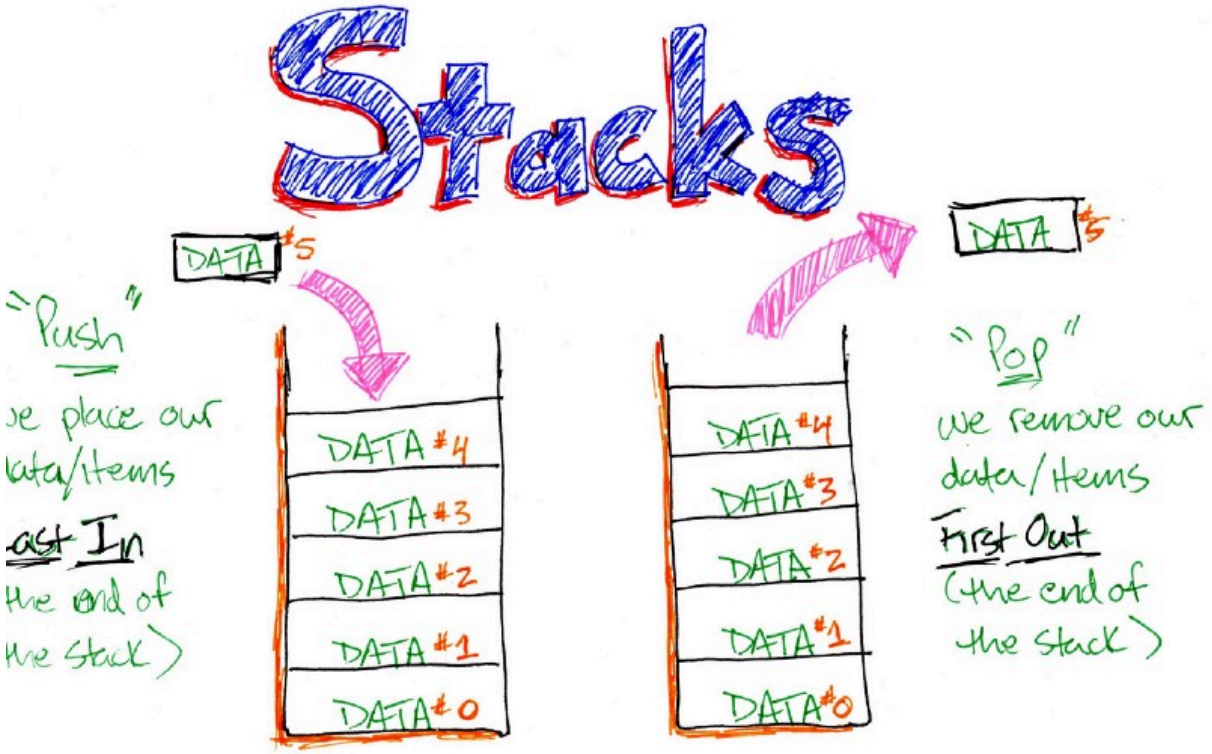
📘 Session 01: Introduction to Stack

✅ What is a Stack?

A **stack** is a linear data structure that follows the **LIFO** principle:

🔁 LIFO = Last In, First Out

> Think of a **stack of plates** – the last plate you put on top is the first one you take off.



📦 Real-World Examples of Stack

- **Undo** operation in editors (Ctrl + Z)
- **Backtracking** in mazes or puzzles
- **Function calls** in recursion
- **Browser history** (Back button)

✅ Why Use a Stack?

Use Case	Example
Undo functionality	Word Processors, IDEs
Expression evaluation	Infix to Postfix, Compiler parsing
Function call handling	Recursion, Stack Frames
Backtracking algorithms	Solving mazes or puzzles

✅ Stack Terminology

Term	Meaning
LIFO	Last In, First Out
push(x)	Insert x at the top of the stack
pop()	Remove item from the top
peek()	View the top element without removing it
isEmpty()	Returns `true` if the stack is empty
Overflow	When pushing into a full stack
Underflow	When popping from an empty stack

✅ Stack Representation Using Array

- Use an **array**: `stack[SIZE]`
- Use an **integer** **`top`** to track top element

Condition	Meaning
top == -1	Stack is empty
top == SIZE - 1	Stack is full

🧠 Stack Step-by-Step Example (SIZE = 5)

🔹 **Initial State:**
`top = -1`, stack is empty

🔹 **After push(10):**

```
| 10 | <- top
```

🔹 **After push(20):**

```
| 20 | <- top
| 10 |
```

🔹 **After pop():**
Removes 20, now top = 0

```
| 10 | <- top
```

🔹 **peek() → 10**
🔹 **isEmpty() → false**

🚫 Stack Error Conditions

Type	When it happens
Overflow	When trying to push but stack is full
Underflow	When trying to pop but stack is empty

💻 Part 2: Stack Implementation in C++ (Using Array and Functions)

```cpp
#include <iostream>
using namespace std;

#define SIZE 5

int stack[SIZE];
```

```
int top = -1;

// Function Declarations
void push();
void pop();
void peek();
void display();
bool isEmpty();  // Returns true if stack is empty

int main() {
    int choice;

    while (true) {
        cout << "\n===== STACK MENU =====\n";
        cout << "1. Push\n";
        cout << "2. Pop\n";
        cout << "3. Peek (Top Element)\n";
        cout << "4. Display Stack\n";
        cout << "5. Exit\n";
        cout << "6. Check if Stack is Empty\n"; s
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                push();
                break;

            case 2:
                pop();
                break;

            case 3:
                peek();
                break;

            case 4:
                display();
                break;

            case 5:
```

```
                cout << "Exiting program.\n";
                return 0;

            case 6:
                if (isEmpty()) {
                    cout << "Stack is EMPTY.\n";
                } else {
                    cout << "Stack is NOT empty.\n";
                }
                break;

            default:
                     cout << "Invalid choice! Please enter a number
between 1 and 6.\n";
        }
    }

    return 0;
}

// Function Definitions

void push() {
    int value;
    if (top == SIZE - 1) {
        cout << "Stack Overflow! Cannot push more elements.\n";
    } else {
        cout << "Enter value to push: ";
        cin >> value;
        top++;
        stack[top] = value;
        cout << value << " pushed into the stack.\n";
    }
}

void pop() {
    if (top == -1) {
        cout << "Stack Underflow! Nothing to pop.\n";
    } else {
        cout << stack[top] << " popped from the stack.\n";
        top--;
```

```
    }
}

void peek() {
    if (top == -1) {
        cout << "Stack is empty.\n";
    } else {
        cout << "Top element is: " << stack[top] << "\n";
    }
}

void display() {
    if (top == -1) {
        cout << "Stack is empty.\n";
    } else {
        cout << "Stack elements (top to bottom):\n";
        for (int i = top; i >= 0; i--) {
            cout << stack[i] << "\n";
        }
    }
}

bool isEmpty() {
    return (top == -1);
}
```

✅ Quiz: Stack Basics

Q1. What is the initial value of `top` when the stack is empty?
→ `-1`

Q2. What condition causes a stack overflow?
→ When `top == SIZE - 1` and you try to push

Q3. What will `isEmpty()` return if the stack has elements?
→ `false`

Q4. After calling `push(10)`, `push(20)`, and `pop()`, what will `peek()` return?
→ `10`

🔹 Part A: Multiple Choice Questions (MCQs)

Choose the correct option.

Q1. What does LIFO stand for in stack terminology?
a) Last In First Out
b) Last Inside First Outside
c) Load In First Out
d) Least Important First Out

Q2. Which of the following is an example of a real-world stack?
a) Queue at a ticket counter
b) Stack of plates
c) Train bogies
d) Car parking

Q3. What is the condition for stack **overflow** in an array of size 5?
a) top == 0
b) top == SIZE
c) top == SIZE - 1
d) top == -1

Q4. What is the value of `top` in an empty stack?
a) 0
b) -1
c) SIZE
d) NULL

Q5. Which function allows you to **look at the top** element without removing it?
a) pop()
b) push()
c) display()
d) peek()

🔹 Part B: True or False

Q6. `push()` removes the top element from the stack.
→ _____

Q7. `pop()` causes underflow if the stack is empty.
→ _____

Q8. `isEmpty()` returns true when `top == -1`.
→ _____

Q9. The stack follows FIFO order.
→ _____

Q10. After `push(10)`, `push(20)`, and one `pop()`, `peek()` will return 10.
→ _____

🔹 Part C: Fill in the Blanks

Q11. In a stack, the element added **last** is the one removed _____.
Q12. `isEmpty()` checks if the stack has _____ elements.
Q13. A stack uses a variable called _____ to track the top element.
Q14. When trying to `pop()` from an empty stack, it leads to _____.
Q15. In a full stack, trying to `push()` an element results in _____.

✅ Answer Key (For Students) : -

Part A:

1. a
2. b
3. c
4. b
5. d

Part B:
6. False
7. True
8. True
9. False
10. True

Part C:

11. first
12. no
13. top
14. underflow
15. overflow