

## Lab Assignment: Two Stacks in a Single Array

### Problem Statement

Design and implement two independent stacks within a single array of fixed size. This approach optimizes memory usage by allowing both stacks to grow towards each other from opposite ends of the array.

### Concept and Strategy

Given an array of size MAX, we define:

- - Stack1 grows from index 0 towards MAX - 1
- - Stack2 grows from index MAX - 1 towards 0

We maintain two pointers:

- top1 for Stack1 (initially -1)
- top2 for Stack2 (initially MAX)

A stack overflow occurs when both stacks attempt to use the same index, i.e., when:  
 $\text{top1} + 1 == \text{top2}$

### Function Signatures and Expected Behavior

- `void push1(int x);`

Pushes an element x into Stack1. Checks for overflow by comparing  $\text{top1} + 1 == \text{top2}$ .  
On success, increments top1 and places x in the array.

- `void push2(int x);`

Pushes an element x into Stack2. Checks for overflow similarly. On success, decrements top2 and places x in the array.

- `int pop1();`

Pops the top element from Stack1. Returns the value and decrements top1. Handles underflow when  $\text{top1} == -1$ .

- `int pop2();`

Pops the top element from Stack2. Returns the value and increments top2. Handles underflow when  $\text{top2} == \text{MAX}$ .

### Time Complexity

All operations in  $O(1)$  time

### Example Workflow

1. Push 10, 20 into Stack1
2. Push 30, 40 into Stack2
3. Pop from Stack1 → returns 20
4. Pop from Stack2 → returns 40