Stack - Introduction, ADT, and Operations

1. Introduction to Stack
A Stack is a linear data structure that follows the **LIFO (Last In, First Out)** principle.
The element that is inserted last is removed first.

Real-life Examples:
- Stack of plates (insert/remove from top).
- Undo/Redo in editors.
- Browser history (Back/Forward navigation).

2. Stack as an Abstract Data Type (ADT)
An Abstract Data Type (ADT) defines **what operations** can be performed but not **how** they are implemented.

Stack ADT Operations:
1. Push(x) → Insert element at top.
2. Pop() → Remove element from top.
3. Peek()/Top() → View top element without removing.
4. isEmpty() → Check if stack is empty.
5. isFull() → Check if stack is full.

3. Operations on Stack (Array Implementation in C)
Below is an example C program demonstrating **Push, Pop, Peek, and Display** operations on a stack.

```
#include <stdio.h>
#define MAX 5   // Maximum size of stack

int stack[MAX];
int top = -1;

void push(int x) {
    if (top == MAX - 1) {
        printf("Stack Overflow! Cannot insert %d\n", x);
    } else {
        stack[++top] = x;
        printf("%d pushed to stack\n", x);
    }
}

void pop() {
    if (top == -1) {
        printf("Stack Underflow! No element to pop\n");
    } else {
        printf("%d popped from stack\n", stack[top--]);
    }
}

void peek() {
    if (top == -1) {
        printf("Stack is empty\n");
    } else {
        printf("Top element is %d\n", stack[top]);
    }
}

void display() {
    if (top == -1) {
        printf("Stack is empty\n");
    } else {
        printf("Stack elements: ");
        for (int i = 0; i <= top; i++) {
            printf("%d ", stack[i]);
        }
        printf("\n");
```

```
    }
}

int main() {
    push(10);
    push(20);
    push(30);
    display();
    peek();
    pop();
    display();
    return 0;
}
```