**Stack Using Array Data Structure**

**Definition**

A stack is a linear data structure that follows the Last In, First Out (LIFO) principle. This means that the last element added to the stack will be the first one to be removed.

**Operations**

1. **Push**: Add an element to the top of the stack.
2. **Pop**: Remove and return the top element from the stack.
3. **Peek/Top**: Return the top element without removing it.
4. **isEmpty**: Check if the stack is empty.
5. **isFull**: Check if the stack is full (relevant for fixed-size stacks).

**Pros and Cons**

**Pros:**

* **Simplicity**: Easy to understand and implement.
* **Direct Memory Access**: As arrays provide direct access to elements, the stack operations are generally very fast.

**Cons:**

* **Fixed Size**: The size of the stack must be defined initially. This can lead to wasted space if the stack is not fully utilized or stack overflow if the stack exceeds its initial size.
* **Memory Usage**: If the stack is not dynamic, memory can be wasted if the initial capacity is too large.

**Applications**

* **Function Call Management**: Used to keep track of function calls and return addresses.
* **Expression Evaluation**: Used in parsing and evaluating expressions (e.g., converting infix to postfix).
* **Undo Mechanism**: Used in software applications to keep track of the history of operations for undo functionality.
* **Backtracking**: Useful in algorithms that require backtracking, such as solving mazes, puzzles, etc.
* **Syntax Parsing**: Used in compilers for syntax parsing of programming languages.