Stack	Queue		
A stack is a linear data structure in which insertions and deletions are allowed only at the end, called the top of the stack.	A queue is defined as a linear data structure that is open at both ends and the operations are performed.		
	Linear	**Circular**	**DeQue**
	Linear or simple queue defines the simple operation of queue in which insertion occurs at the rear of the list and deletion occurs at the front of the list.	A circular queue is one in which the insertion of a new element is done at very first location of the queue if the last location of the queue is full.	In Double Ended Queue insert and delete operation can be occur at both ends that is front and rear of the queue.
	Possible to enter new items from the rear end and remove the items from the front.	Possible to enter and remove elements from any position.	
	Requires more memory.	Requires less memory.	
	Less efficient.	More efficient.	
When we define a stack as an ADT (Abstract Data Type), then we are only interested in knowing the stack operations from the user point of view. Means we are not interested in knowing the implementation details at this moment. We are only interested in knowing what type of operations we can perform on stack.	When defining a queue as an Abstract Data Type (ADT), the focus is on the operations from a user's perspective, abstracting away implementation details.		
It follows LIFO (Last in First Out) principle.	It follows FIFO (First in First Out) principle.		

Applications	
Application of Stack Data Structure: • **Function calls and recursion:** When a function is called, the current state of the program is pushed onto the stack. When the function returns, the state is popped from the stack to resume the previous function's execution. • **Undo/Redo operations:** The undo-redo feature in various applications uses stacks to keep track of the previous actions. Each time an action is performed, it is pushed	**Application of Queue Data Structure:** • CPU scheduling, Disk Scheduling. • When data is transferred asynchronously between two processes. The queue is used for synchronization. For example: IO Buffers, pipes, file IO, etc. • Handling of interrupts in real-time systems. • Call Center phone systems use Queues to hold people calling them in order.

onto the stack. To undo the action, the top element of the stack is popped, and the reverse operation is performed.

- **Expression evaluation:** Stack data structure is used to evaluate expressions in infix, postfix, and prefix notations. Operators and operands are pushed onto the stack, and operations are performed based on the stack's top elements.

- **Browser history:** Web browsers use stacks to keep track of the web pages you visit. Each time you visit a new page, the URL is pushed onto the stack, and when you hit the back button, the previous URL is popped from the stack.

- **Balanced Parentheses:** Stack data structure is used to check if parentheses are balanced or not. An opening parenthesis is pushed onto the stack, and a closing parenthesis is popped from the stack. If the stack is empty at the end of the expression, the parentheses are balanced.

- **Backtracking Algorithms:** The backtracking algorithm uses stacks to keep track of the states of the problem-solving process. The current state is pushed onto the stack, and when the algorithm backtracks, the previous state is popped from the stack.

Time Complexity			
push()	O(1)	enQueue()	O(1)
pop()	O(1)		
peek()	O(1)	deQueue()	O(1)
isFull()	O(1)		
isEmpty()	O(1)	peek()	O(1)
size()	O(1)		
Incremental Strategy: push() [Amortized Time]	O(n)	isFull()	O(1)
Doubling Strategy: push() [Amortized Time]	O(1)	isEmpty()	O(1)
		size()	O(1)

Space Complexity			
n push operations	O(n)	n enQueue operations	O(n)