**Data Structure - Stack**

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

**Mainly the following three basic operations are performed in the stack**

* ***LIFO Structure: In a stack, the last element added (pushed) onto the stack is the first one to be removed (popped). This is known as Last In, First Out (LIFO).***
* **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
* **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
* **Peek or Top:** Returns top element of stack.
* **isEmpty:** Returns true if stack is empty, else false.

**Algorithm: PUSH Operation in a Stack**

1. Start
2. Input the stack **S**, the top position **top**, the maximum size of the stack **maxSize**, and the element **elem** to be pushed.
3. Check if the stack is full:

If **top** is equal to **maxSize - 1**, print **"Stack Overflow"** and exit.

1. Increment the **top** position by 1.
2. Insert the element at the **top** position:

**S[top] = elem**

1. End

**Pseudocode:**

**void push(int S[], int \*top, int maxSize, int elem)**

**{**

**if (\*top == maxSize - 1)**

**{**

**printf("Stack Overflow\n");**

**return;**

**}**

**\*top = \*top + 1;**

**S[\*top] = elem;**

**}**

**Algorithm: POP Operation in a Stack**

1. Start
2. Input the stack **S** and the top position **top**.
3. Check if the **stack is empty**:
4. If **top** is **-1**, print **"Stack Underflow"** and exit.
5. Retrieve the element at the **top** position.
6. Decrement the top position by 1.
7. End

**Pseudocode:**

**int pop(int S[], int \*top)**

**{**

**if (\*top == -1)**

**{**

**printf("Stack Underflow\n");**

**return -1;**

**}**

**int elem = S[\*top];**

**\*top = \*top - 1;**

**return elem;**

**}**