Algorithm - Stack

1. **PUSH**(stack, top, element)
    a. if top >= MAX_SIZE - 1
        print "Stack Overflow"
    b. else
        top = top + 1
    c. stack[top] = element
    d. print element, "pushed into stack"

2. **POP**(stack, top)
    a. if top == - 1
        print "Stack Underflow"
    b. else
        element = stack[top]
    c. top = top - 1
    d. print element, "popped from stack"

3. **IS_EMPTY**(top)
    a. if top == -1
        return True
    b. else
        return False

4. **IS_FULL**(top)
    a. if top >= MAX_SIZE - 1
        return True
    b. else
        return False

Implementation of Stack

1. Initialize stack with a certain MAX_SIZE = n
2. Perform stack operations:
    a. Check if the stack is full using **IS_FULL()**.
        i. if top >= MAX_SIZE - 1
        return True
        ii. else
        return False
    b. Push an element into the stack using **PUSH()**.
        i. if top >= MAX_SIZE - 1
        print "Stack Overflow"
        ii. else
        top = top + 1
        iii. stack[top] = element
        iv. print element, "pushed into stack"

c. Pop an element from the stack using **POP()**.
    i. f top == - 1
       print "Stack Underflow"
    ii. else
       element = stack[top]
    iii. top = top - 1
    iv. print element, "popped from stack"
d. Check if the stack is empty using **IS_EMPTY()**.
    i. if top == -1
       return True
    ii. else
       return False