



# The Stack

Neha Shah



# Definition

It is an ordered collection of an elements in which insertion and deletion takes place at one end only called as Top of the Stack.

It is linear Data structure.

4	50
3	40
2	30
1	20
0	10

Stack works on the principle of Last In First Out(LIFO).

Last element inserted will be deleted first from the stack.



# TOP

- It is the position of elements in the stack.
- When there are no elements in the stack top of the stack will be -1.



# Stack Implementation

- There are two ways to implement stack
  1. Static Implementation: In static, we use array
  2. Dynamic Implementation: in dynamic we use linked list



# Operations on stack

- There are two operations we can perform on stack
  - Push
  - Pop
- Push : Inserting an element into stack
- Pop: Deleting an element from stack



# Push Algorithm

- Before inserting an element into the stack we must check for stack full condition that is if there is space for element then only we can insert new element
- As we are implementing stack using an array
- Algorithm:
  1. Check whether stack is full or not  $[top == \text{Maxsize} - 1]$
  2. If stack is full print message stack is full
  3. Otherwise, increment top by one position  $[top++]$
  4. Take an element from the user and insert in at the  $top[\text{stack}[top] = \text{element}]$



# Pop Algorithm


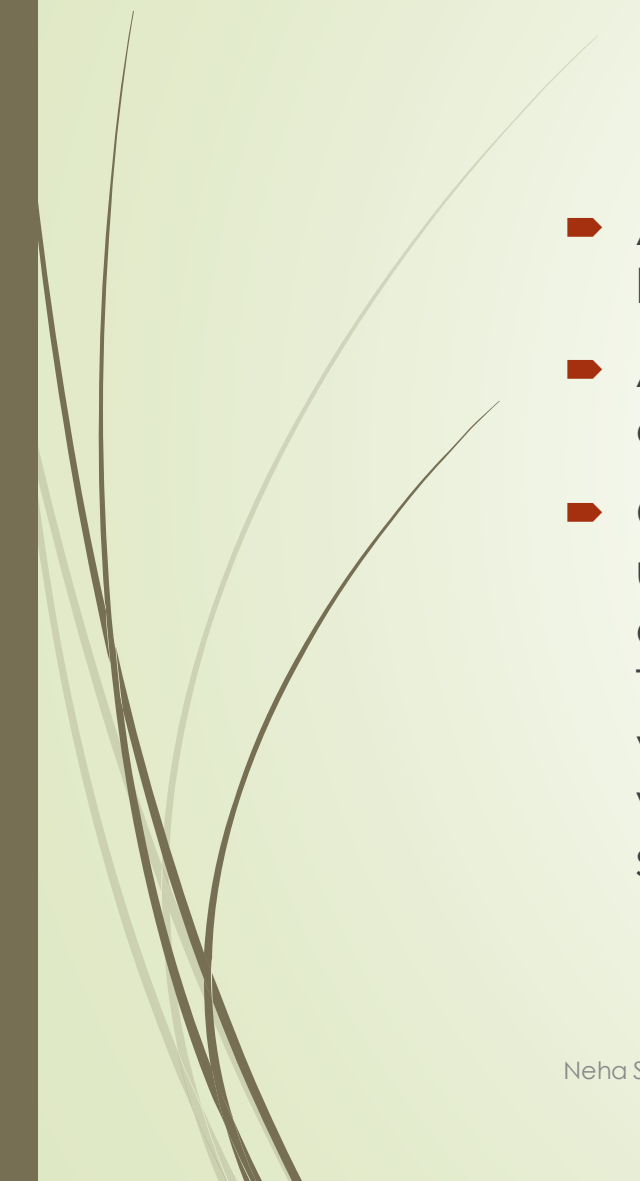
- Before deleting an element from the stack we must check for stack empty condition that is if there is an element in the stack then only we can deletion operation is possible
- As we are implementing stack using an array
- Algorithm:
  1. Check whether stack is empty or not  $[top == -1]$
  2. If stack is full print message stack is empty
  3. Otherwise, delete an element from the stack  $[element = stack[top]]$
  4. Decrement top by one position  $[top--]$



# Applications of Stack

- Expression Evaluation
- Recursion
- Towers of Hanoi



- 
- 
- All allocation made by `malloc()`, `calloc()` or `realloc()` are stored on the heap, while all local variables are stored on the stack.
  - All global and static variables are stored in the data segment, while constants are stored in the code segment.
  - Global variables can be in a couple places, depending on how they're set up — for example, `const` globals may be in a read-only section of the executable. "Normal" globals are in a read-write section of the executable. They're not on the heap or the stack at all. Pointers are just a type of variable, so they can be wherever you want them to be (on the heap if you `malloc()` them, on the stack if they're local variables, or in the data section if they're global).