Stack :- A stack is a linear data structure that follows LIFO (Last-In-First-Out) principle. Stack has one end, whereas queue has two ends (front and rear).

A stack is a container in which insertion and deletion can be done from the end (one) known as the top of the stack.

A stack is an Abstract Data Type with a pre-defined capacity, which means that it can store elements of limited size.

Operations on the stack :-

1). push ( ) :- When we insert an element in a stack then the operation is known as push. If stack is full overflow condition occurs.

2). pop ( ) :- when we delete an element from stack, the operation is called as pop ( ). If stack is empty means no element exists in the stack, this state is known as an underflow state.

3). peek ( ) :- It returns the element at a given position.

4). count ( ) :- It returns the total number of elements available in a stack.

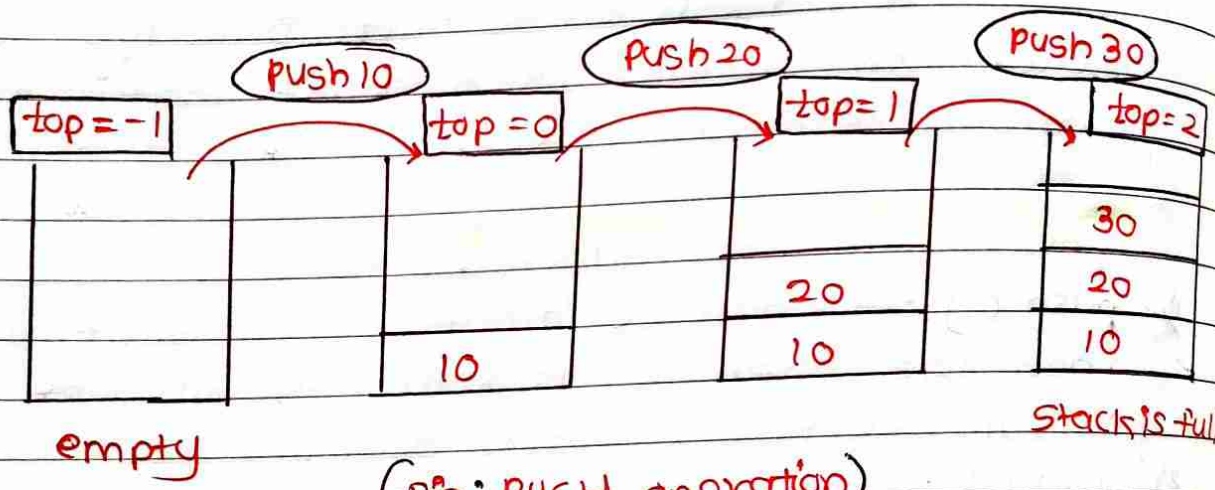
5). change ( ) :- It changes the element at the given position.

6). display ( ) :- It prints all the elements available in the stacks.

PUSH operation :-

Steps - Before inserting an element in the a stack, we check whether the stack is full.

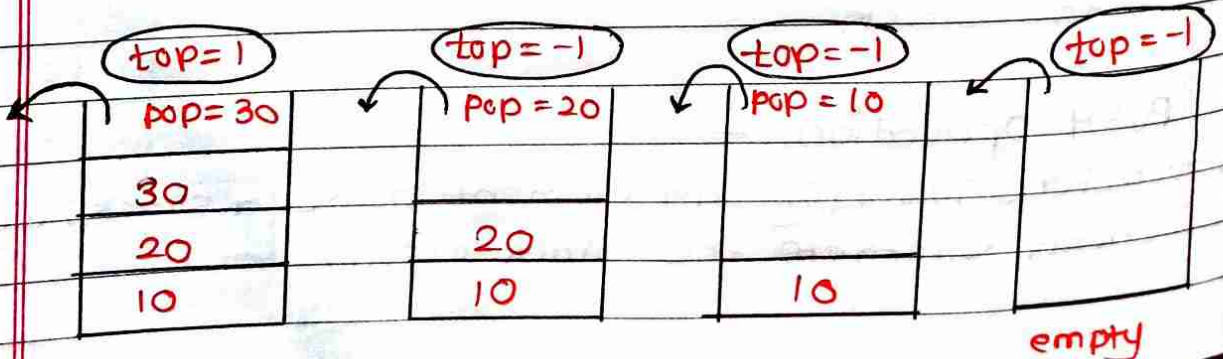
- If we try to insert element in a stack, and the stack is full, then overflow condition occurs.
- when we initialised a stack, we set the value of top as -1 to check that stack is empty
- The elements will be inserted until we reach the max size of the stack. top = top + 1.



(fig: PUSH operation)

POP Operation :-

- Before deleting the element from the stack, we check whether the stack is empty.
- If we try to delete the element from empty stack, then underflow condition occurs.
- first access the element which is pointed by top.
- once the top operation is performed, top is decremented by 1 i.e. top = top - 1.



Applications of stack :-

1) Recursion :- The recursion means that the function is calling itself again. To maintain the previous states, the compiler creates a system stack in which all previous records of function are maintained.

2) DFS (Depth first search) :- This search is implemented on a graph, graph uses stack d.s.

3) Backtracking :- If we have to create a path to solve maze problem, If we are moving in particular path and we realise that we come on the wrongway. In order to come at beginning of the path to create a new path, we use stack d.s.

4) Memory management :- The stack manages the memory. The memory is assign in the contiguous memory blocks.

Algo :- push operation :-

```
begin
  if top = n then stack full
  top = top + 1
  stack(top) := item ;
end
```

Time Complexity : O(1)

pop operation :-

```
begin
  if top = 0 then empty
  item := stack(top) ;
  top = top - 1 ;
end.
```

Time complexity : O(1)