October 7

Stack - Linear Data structure which follows a particular order in which operations are performed.

1) LIFO (last In first out)
2) FIFO (first in first out)

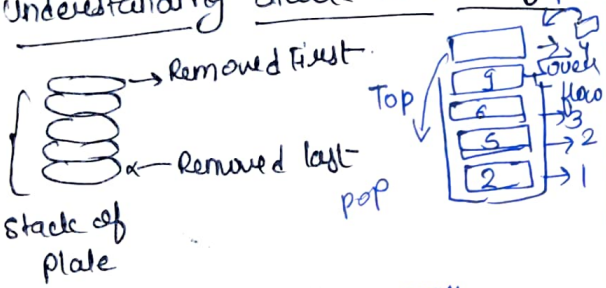
Operations in stacks -

(a) push - add an item + If stack is full → overflow condn

(b) pop - removes an item (items are popped in reverse order in which they are pushed) + If stack is empty → underflow condition (d) Isempty / Is full

(c) Peek - Getting Top most item.

Understanding stack Practically



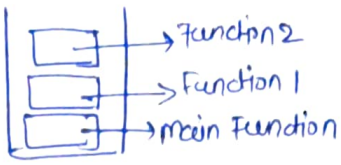
Implementation → using array
→ using linked list

structure to represent a stack

```
struct stack
{ int top;
  unsigned capacity;
  int * array;
}
```

Applications of stack

- Used in Function calls
- Infix to postfix conversion
- Parenthesis matching & more



main function will call each function.

Stack as abstract Data type

→ In order to create a stack we need a pointer to the topmost element along with other elements which are stored inside the stack.

Stack using Array Data structure

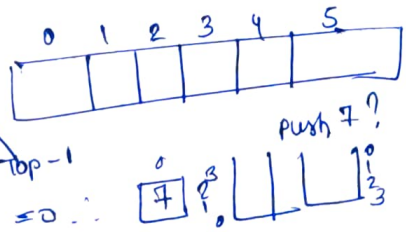
→ Fixed size array creation

→ Top Element

```
struct stack {
    int size;
    int top;
    int arr;
};
```

```
struct stack s;
s.size = 80;
s.top = -1
s.arr = (int *) malloc (sizeof(int) * s.size);
malloc (s.size * sizeof (int));
```

Example



Now, Top = 0 ∴

↓

Denotify Till when stack is filled.

Above we are performing push & pull operation on stack using array.