DATA STRUCTURES

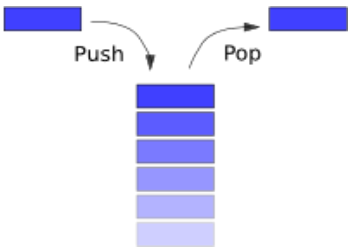
UNIT 2

STACK

A Stack is an ordered list in which all insertions and deletions are made at one end, called Top.
The last element to be inserted into the stack will be the first to be removed.
So, the stacks are referred as Last In First Out (LIFO)

Or

A stack is a particular kind of abstract data type or collection in which the operations addition of an entity to the collection is known as push and removal (deletion) of an entity to the collection is known as pop.
The relation between the push and pop operations is the stack's Last-In-First-Out (LIFO) data structure.



Real life examples of stacks are

1. Stack of books
2. Stack of clothes
3. Stack of plates

SIMPLE REPRESENTATION OF A STACK USING ARRAY

Given a stack S = (a[1], a[2],.....a[n]) then we say that a1 is the bottom most element and element a[n] is on the top.

The operations on stack is

1. Push (Insertion)
2. Pop (Deletion)
3, Peek (Top most Element)
4. Overflow (Full)
5. Underflow (Empty)

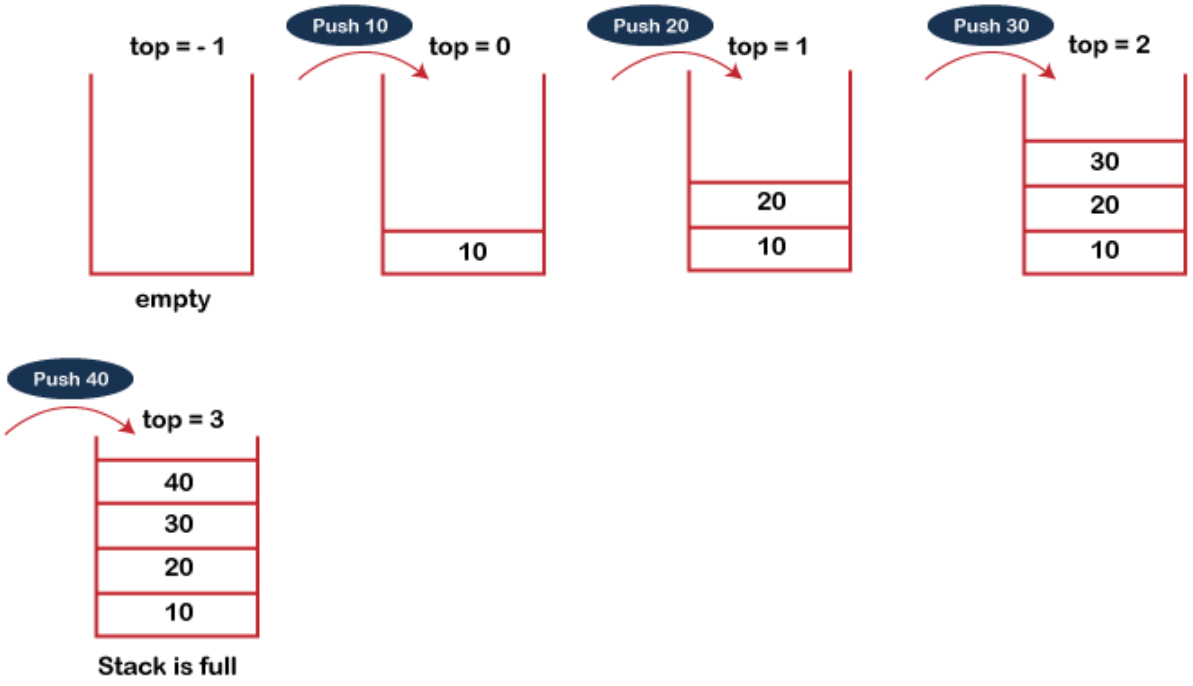
Stack – Abstract Data Type

A stack 'S' is an abstract data type (ADT) supporting the following three methods:
push(n) : Inserts the item n at the top of stack
pop(n) : Removes the top element from the stack and returns that top element. An error occurs if the stack is empty.
peek(n) : Returns the top element and an error occurs if the stack is empty.

PUSH OPERATION

Push operation inserts an element onto the stack.
Let's represented the stack by means of array.
An attempt to push an element onto the stack, when the array is full, causes an overflow.

Push operation involves :
- Check whether the array is full before attempting to push another element. If so, halt the execution.
- Otherwise increment the top pointer.
- Push the element onto the top of the stack.



Overflow

The steps involved in the PUSH operation :

- Before inserting an element in a stack, check whether the stack is full.
- If we try to insert the element in a stack, and the stack is full, then the overflow condition occurs.
- When initialize the stack, set the value of top as -1 to check that the stack is empty.
- When the new element is pushed in a stack.
- The value of the top gets incremented, i.e., top = top + 1, and the element will be placed at the new position of the top.
- The elements will be inserted until we reach the max size of the stack.

Algorithm for PUSH Operation

```
Algorithm Add(item)
// Push an element onto the stack. Return true if successful;
// else return false. item is used as an input.
{
    if (top ≥ n − 1) then
    {
        write ("Stack is full!"); return false;
    }
    else
    {
        top := top + 1; stack[top] := item; return true;
    }
}
```

Or

```
procedure add(item : items);
{add item to the global stack stack;   top is the current top of stack and n is its maximum size}
begin
   if top = n then stackfull;
   top := top+1;
   stack(top) := item;
end: {of add}
```

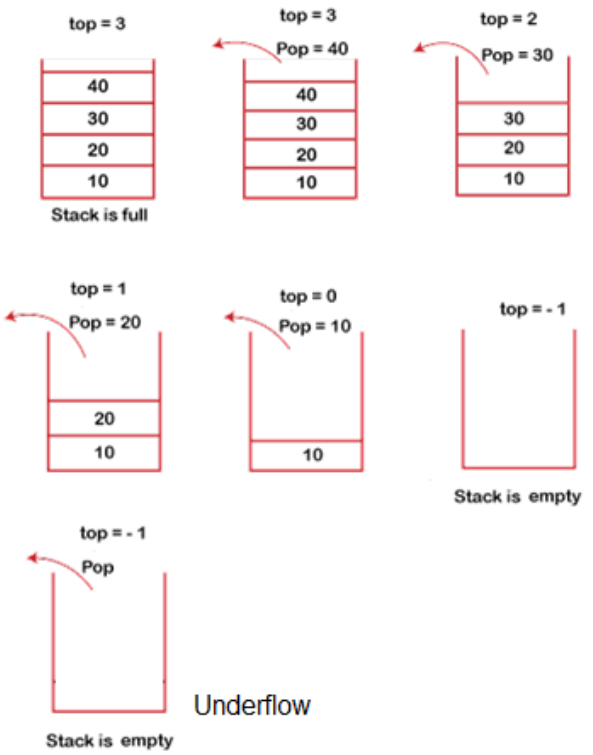
POP OPERATION

POP operation removes an element from the stack.
An attempt to pop an element from the stack, when the array is empty, causes an underflow.

Pop operation involves :
- Check whether the array is empty before attempting to pop another element. If so, halt execution.
- Decrement the top pointer.
- Pop the element from the top of the stack.

The steps involved in the POP operation :

- Before deleting the element from the stack, we check whether the stack is empty.
- If we try to delete the element from the empty stack, then the underflow condition occurs.
- If the stack is not empty, we first access the element which is pointed by the top
- Once the pop operation is performed, the top is decremented by 1, i.e., top=top-1.



Algorithm for POP Operation

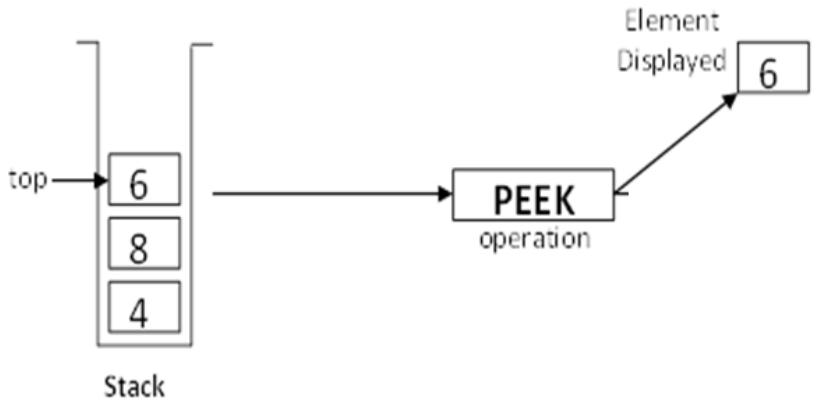
```
Algorithm Delete(item)
// Pop the top element from the stack. Return true if successful
// else return false. item is used as an output.
{
    if (top < 0) then
    {
        write ("Stack is empty!"); return false;
    }
    else
    {
        item := stack[top]; top := top − 1; return true;
    }
}
```

Or

```
procedure delete(var item : items);
{remove top element from the stack stack and put it in the item}
begin
   if top = 0 then stackempty;
   item := stack(top);
   top := top-1;
end; {of delete}
```

Peek Operation:

- Returns the item at the top of the stack but does not delete it.
- This can also result in underflow if the stack is empty.



Algorithm for PEEK Operation

```
PEEK(STACK, TOP)
        BEGIN
/* Check, Stack is empty? */
if (TOP == -1) then
        print “Underflow” and return 0.
else
        item = STACK[TOP]      / * stores the top element into a local variable */
        return item            / * returns the top element to the user */
        END
```

Applications of Stack

1. Towers of Hanoi
2. Reversing a string
3. Balanced parenthesis
4. Recursion using stack
5. It is very useful to evaluate arithmetic expressions. (Postfix Expressions)
6. Infix to Postfix Transformation
7. It is useful during the execution of recursive programs
8. A Stack is useful for designing the compiler in operating system to store local variables inside a function block.
9. A stack (memory stack) can be used in function calls including recursion.
10. Backtracking.