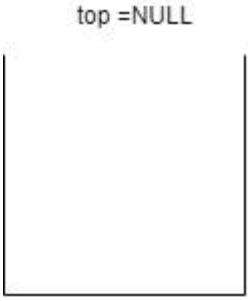
Dynamic Implementation of Stack

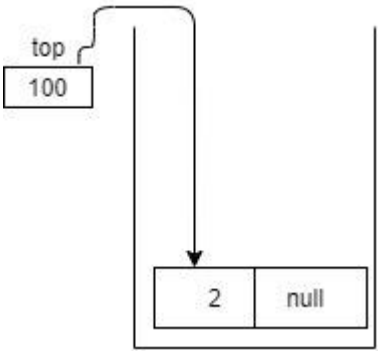
Example of PUSH

Suppose we have to perform operations push(2), push(5), push(7).

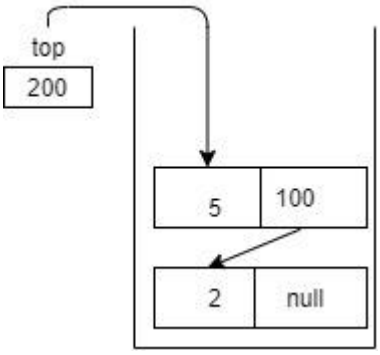
- in starting stack was empty and top is pointing to null.



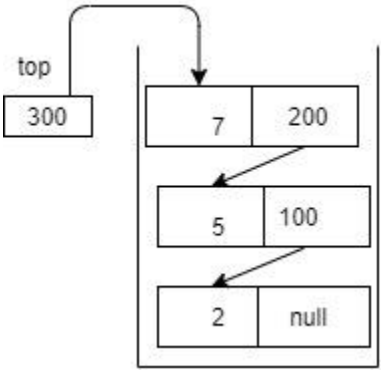
- To insert 2 we will first create a new node and set its data part as 5, lets suppose new node memory address is 100. since we are doing operation at the head end so top should now point to newly created node (i.e address 100) and the address part of node is set to null.



- To insert 5 in the stack we again create a new node, data part is set to 5 and lets suppose new node memory address is 200 now since this new node should point to previous node and top should now point to new node, we will copy the address contained in top to address part of new node and set top pointing to memory address of new node.



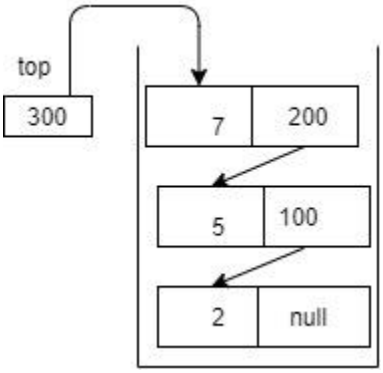
- And similarly 7 is inserted (lets suppose new node memory address is 300).



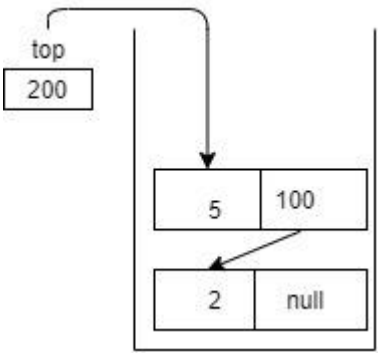
In this linked list implementation of stack we don't need to check if memory overflow is happening while inserting data in stack, because we have not fixed the size of stack, we are creating node dynamically based on demand at runtime, so no issue of overflow arises.

2. pop() : We use this method to delete element from the stack.
Example:

Suppose we have to perform pop() operations on the following stack.



- On pop(), node containing 7 will be deleted and top now will point to next node i.e node containing data 5. So top will point to address 200. So after the first pop() operation, stack will look like-



3. display() : This method is used to print the elements of stack.