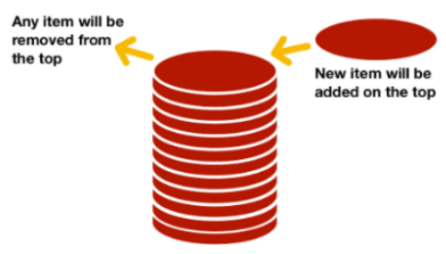
Information

Stack is a linear data structure which is a collection of elements which are inserted or deleted according to the LIFO rule i.e. Last-In-First-Out. Take the example of a stack of disks which are placed one on top of another we keep on adding new disks to the top and when we need to take one, we take the topmost one very similar to how a stack works.



Operations performed with a Stack:

- Push() - To insert data into the stack
- Pop() - To remove/delete data from the stack
- isEmpty() - To check whethet a stack is empty or not
- isFull() - To check whethet a stack is full or not
- StackTop() - To find what is at the top of the stack
- Display() – To print elements in the stack

1.

Implement Stack and its functions using Array. Consider the given example pseudo-code for push and pop using an array.

Watch the following video for more details: https://www.youtube.com/watch?v=rS-ZKTqwi90&ab_channel=NesoAcademy

```
PUSH(S, x)
   S.top = S.top+1
   if S.top > S.size
        Error "Stack Overflow"
   else
        S[S.top] = x
```

```
POP(S)
   if IS_EMPTY(S)
        Error “Stack Underflow”
   else
        S.top = S.top-1
        return S[S.top+1]
```

*Note: The maximum size of the stack should be passable as an argument to the constructor. In addition, the **Stack** class must include the following methods with the exact method names:*

- *push()*
- *pop()*
- *isEmpty()*
- *isFull()*
- *stackTop()*
- *display() - Print the elements from the top of the stack to the bottom, separated by spaces*

2.

Implement Stack and its functions using LinkedList. Consider the given example pseudo-code for push and pop using a linked list.

Watch the following video for more details: https://www.youtube.com/watch?v=0-kkDfCOXOI&ab_channel=NesoAcademy

```
PUSH(S, n)
  if IS_EMPTY(S) //stack is empty
       S.head = n //new node is the head of the linked list
       S.top = n //new node is the also the top
  else
       S.top.next = n
       S.top = n
```

```
POP(S)
  if IS_EMPTY(S)
       Error "Stack Underflow"
  else
       x = S.top.data
       if S.top == S.head //only one node
            S.top = NULL
            S.head = NULL
       else
            tmp = S.head
            while tmp.next != S.top //iterating to the node previous to top
                 tmp = tmp.next
            tmp.next = NULL //making the next of the node null
            S.top = tmp //changing the top pointer
       return x
```

*Note: The **Stack** class must include the following methods with the exact method names:*

- *push()*
- *pop()*
- *isEmpty()*
- *stackTop()*
- *display() - Print the elements from the top of the stack to the bottom, separated by spaces*