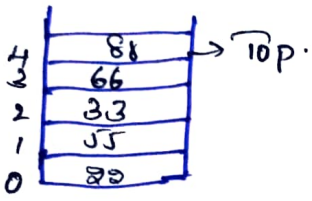
J. Lokesh
192311214

Assignment-2

* Perform the following operations using Stack. Assume the size of the Stack is 5 and having a value of 22, 55, 33, 66, 88 in the stack from 0 position to size-1. Now perform the following operation : 1.) Insert the element in the Stack 2.) pop() 3.) pop() 4.) push(90) 5.) push(36) 6.) push(11), 7.) push(4) 8.) pop(). Draw the diagram of stack and illustrate the above operations and identify Initial Stack.



Operations :-

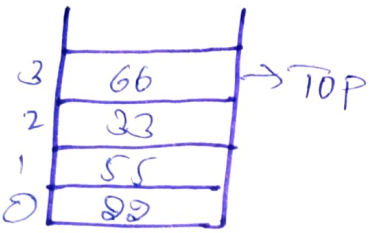
1.) Insert the elements in the Stack :-

The Stack is already initialized with the elements.

[22, 55, 33, 66, 88]

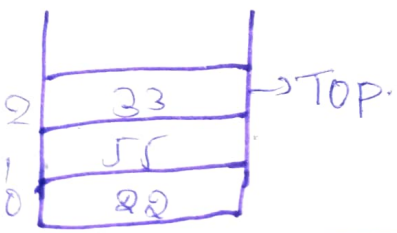
2.) pop() :- Remove the top element (88)

Stack after pop() :



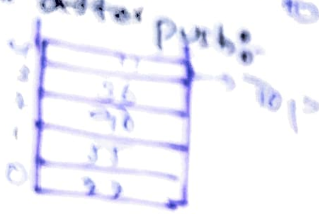
3.) Pop() :- Remove the next top element (66)

Stack after pop() :-

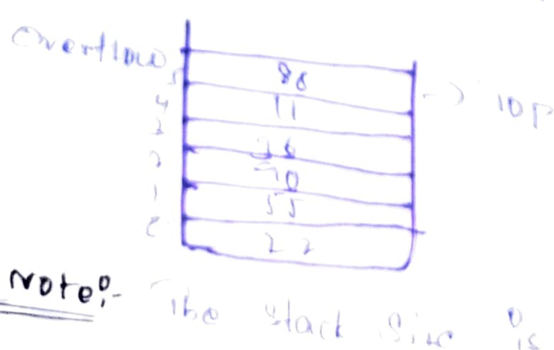


6.) Push(36): Add 36 to the stack
Stack after Push(36):

7.) Push(11): Add 11 to the stack
Stack after Push:



8.) Push(86):- The Stack is now full, so pushing another shouldn't be allowed & it should raise an overflow. However, if we assume the problem statement implement we have capacity, we can proceed.

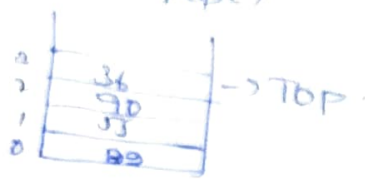


Note:- The stack size is exceeded, indicating an overflow c...

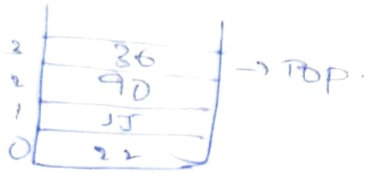
9.) pop(): Remove the top element (86), assuming overflow handling.

(b) Pop(): Remove the top element (11)

Stack after pop()



Final Stack State:



IDENTIFICATION of the top:- The top of the Stack is Currently at index 3, with the value 36

Conclusion:-

* The operation on the Stack was performed as Specified and the current top element is 36 at index 3.

* The Stack initially has elements, which were then popped, and new elements were pushed.

2) Develop an algorithm to delet duplicate elements in array unsorted array using linear Search. Determine the time Complexity and discuss how you should optimize this process.

Sol:- To delect duplicate elements in an unsorted array using linear Search you can use a brute-force approach that involves comparing each element with every other element in the array. Hence, a simple implementation in pseudocode:-

Pseudo Code:-

```
function findDuplicante(arr):
    duplicante = [ ]
    n = length(arr)
    for i=0 to n-1:
        for j=i+1 to n-1:
            if arr[i]= arr[j] and arr[i] not in duplicantes:
duplicates.append(arr[j])
    return duplicates.
```

Explanation:-

* Create an empty list duplicates to store duplicate elements.
* Iterate through each element arr[i] in the array
* For each arr[i], Compare it with every subsequent element arr[j].
* If arr[i]= =arr[j] and the element is not already in the duplicate list, add it to duplicates.
* After both loops complete, return the list of duplicates.

TIME COMPLEXITY:-

The time Complexity of this brute-force approach is $(O(n^2))$, where (n) is the number of elements in the array. This is because, for each element, the algorithm compares it with every other element in the array.

1.> Using A Hash Set :-

Pseudocode:-

```
function findDuplicates(arr):
    seen = set()
    duplicates = []
    for element in arr:
        if element in seen:
            duplicates.append(element)
        else:
            seen.add(element)
    return duplicates
```

Explanation:

* Set Seen:- A set to store elements as we iterate through that array.
* Check for Duplicate:- For each element, check if it is already in the set seen. If it is, add it to the duplicate list because it has been identified as a duplicate.

Result:-

After iterating through the entire array, the function returns the duplicates list, which contains all elements that were found more than once in the input array.