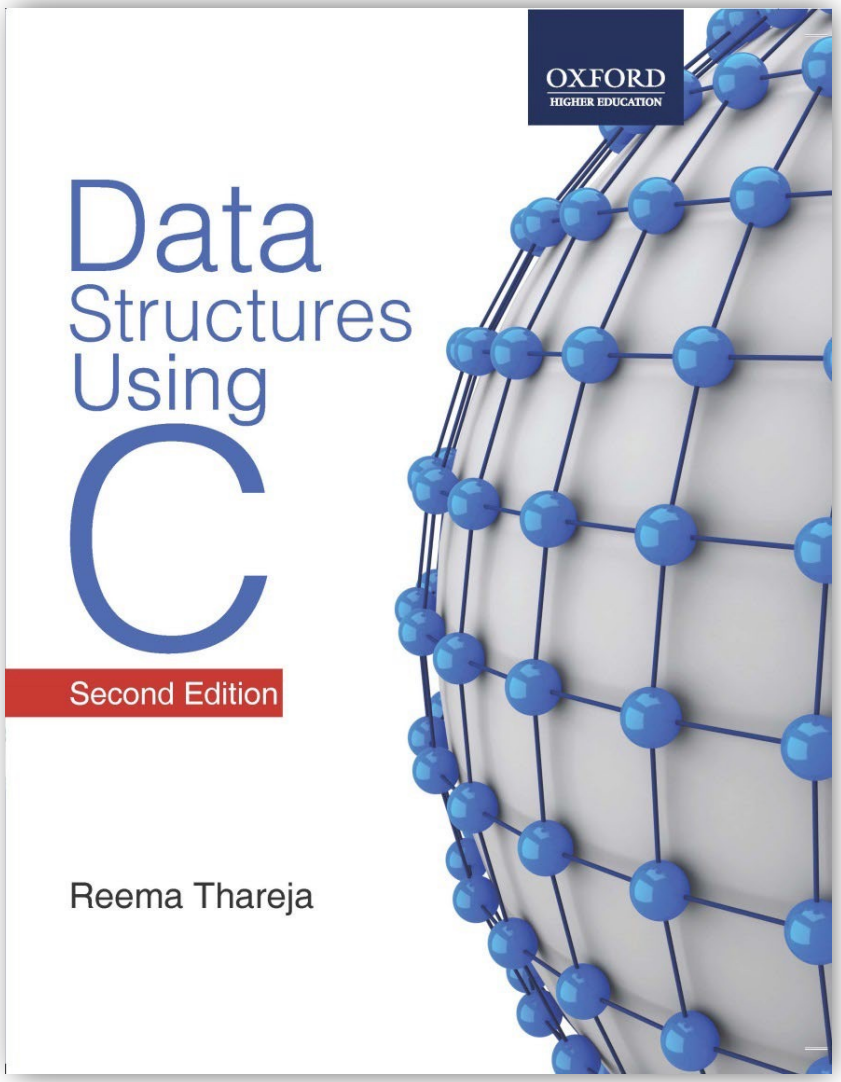


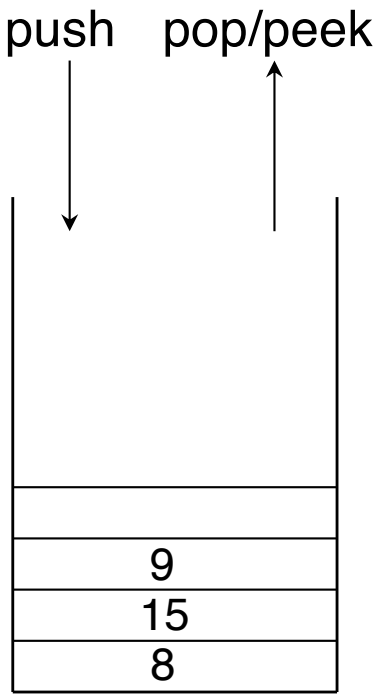
Data Structures Using C, 2e

Reema Thareja

Chapter 7

Stacks

Stack: A Linear Data Structure



🔷 **Definition**: A stack is a linear data structure that follows the **Last In, First Out (LIFO)** principle.

- Items can be added and removed **only from the top**.
- Direct access, insertion, or deletion at arbitrary positions is **not allowed**.

🔷 **Operations**:

✅ **push(ItemType i)** – Adds an item to the **top** of the stack.

✅ **pop()** – Removes and returns the item from the **top**.

✅ **peek()** – Returns the item at the **top** without removing it.

✅ **isEmpty()** – Checks if the stack is **empty**. underflow

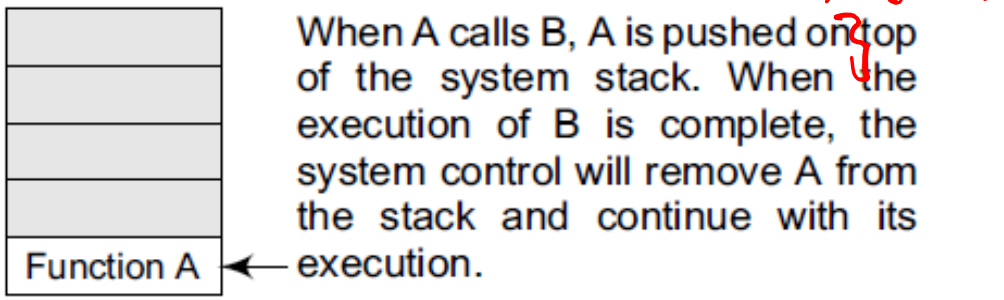
✅ **isFull()** – Checks if the stack is **full** (for fixed-size stacks). overflow

🔷 **Typically take O(1)** operations.

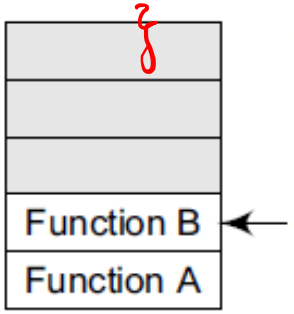
push(8); push(15); push(9); pop() - returns 9

Stacks

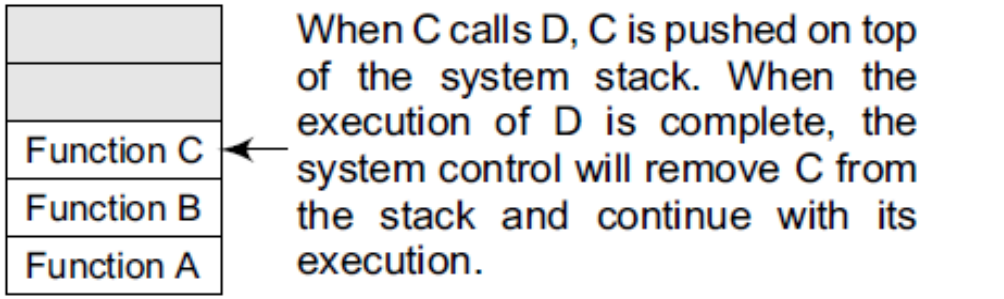
- In order to keep track of the returning point of each active function, a special stack called system stack or call stack is used.
- Whenever a function calls another function, the calling function is pushed onto the top of the stack so that after the called function gets executed, the control is passed back to the calling function.



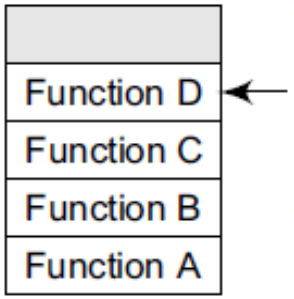
When A calls B, A is pushed on top of the system stack. When the execution of B is complete, the system control will remove A from the stack and continue with its execution.



When B calls C, B is pushed on top of the system stack. When the execution of C is complete, the system control will remove B from the stack and continue with its execution.

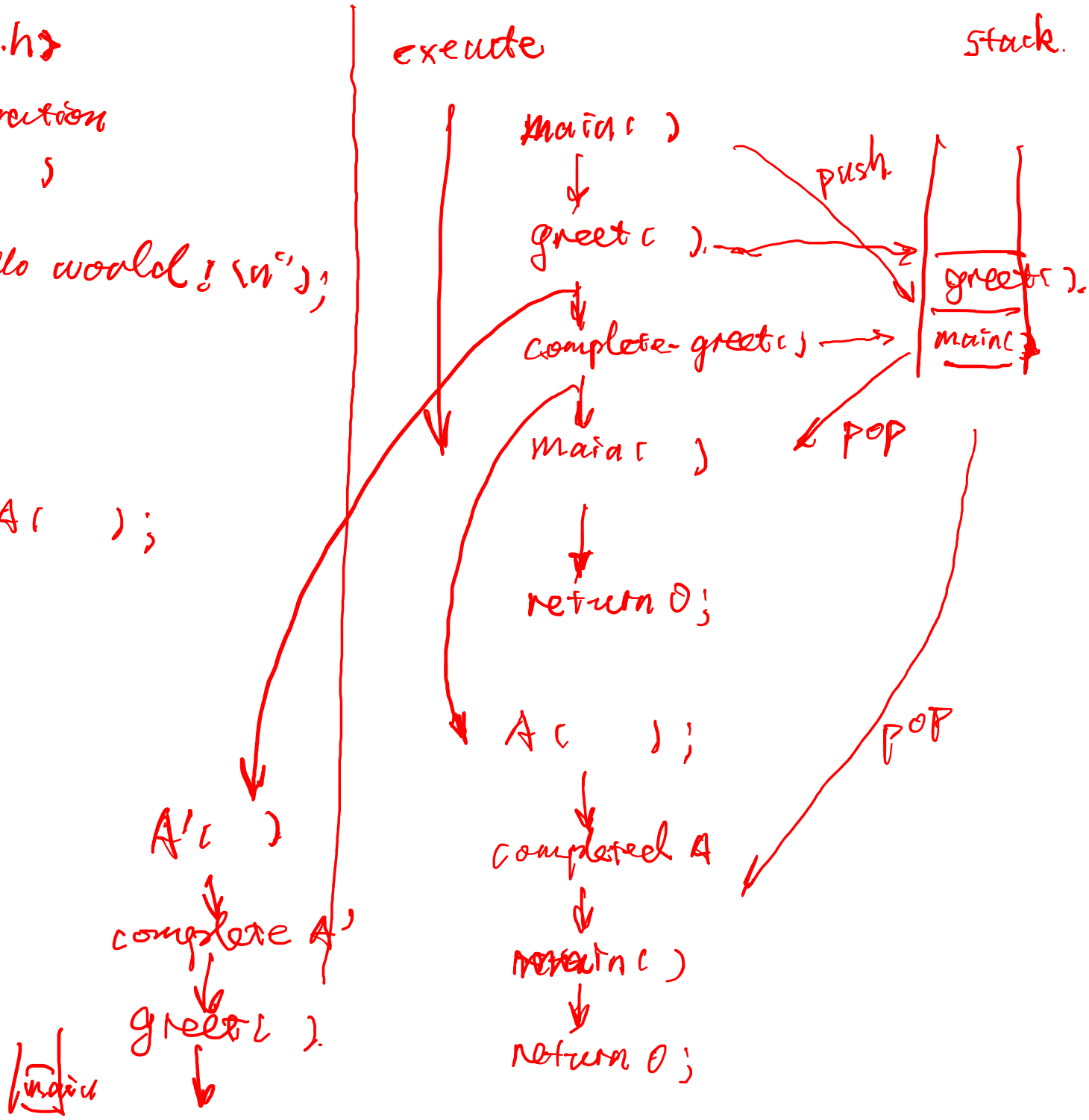


When C calls D, C is pushed on top of the system stack. When the execution of D is complete, the system control will remove C from the stack and continue with its execution.



When D calls E, D is pushed on top of the system stack. When the execution of E is complete, the system control will remove D from the stack and continue with its execution.

```
#include <stdio.h>
// Function declaration
void greet();
void A();
int main() {
    greet();
    A();
    return 0;
}
void greet() {
    printf("Hello world!\n");
}
```

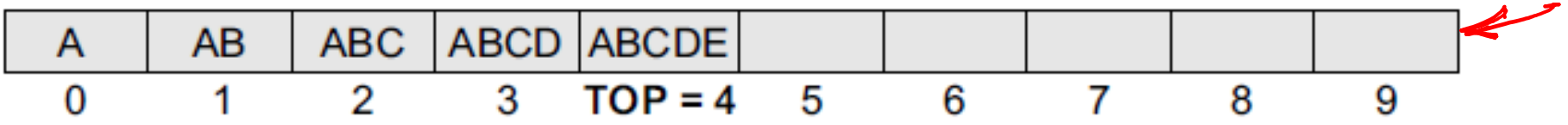


Stack Implementation

- Stack can be implemented in two ways:
  - Static implementation – i.e. arrays
  - Dynamic implementation – i.e. linked list

Array Representation of Stacks

- In the computer's memory, stacks can be represented as a linear array.
- Every stack has a variable TOP associated with it.
- TOP is used to store the address of the topmost element of the stack. It is this position from where the element will be added or deleted.
- There is another variable MAX which will be used to store the maximum number of elements that the stack can hold.
- If TOP = NULL, then it indicates that the stack is empty and if TOP = MAX -1, then the stack is full.



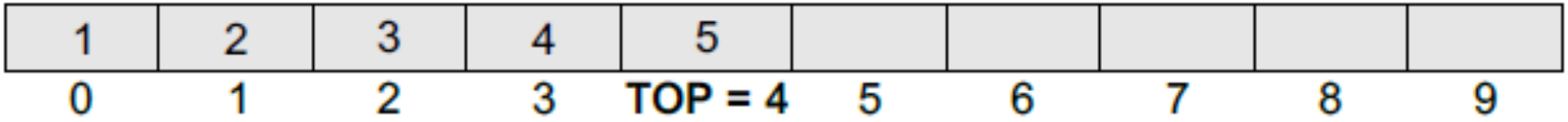
Push Operation

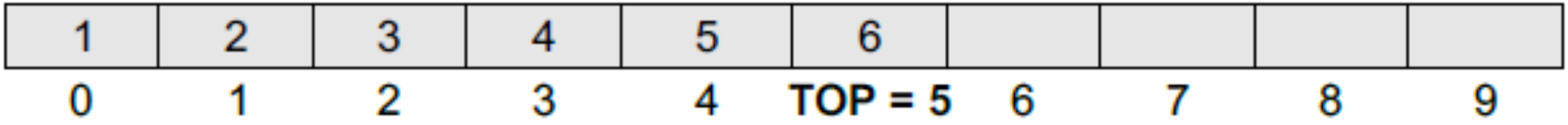
- The push operation is used to insert an element into the stack.
- The new element is added at the topmost position of the stack.
- However, before inserting the value, we must first check if TOP=MAX-1, because it means the stack is full and no more insertions can be done.
- If an attempt is made to insert a value in a stack that is already full, an OVERFLOW message is printed.

Push Operation

```
Algorithm to PUSH an element in a stack

Step 1: IF TOP = MAX-1, then
            PRINT "OVERFLOW"
            GOTO Step 4
        [END OF IF]
Step 2: SET TOP = TOP + 1
Step 3: SET STACK[TOP] = VALUE
Step 4: END
```





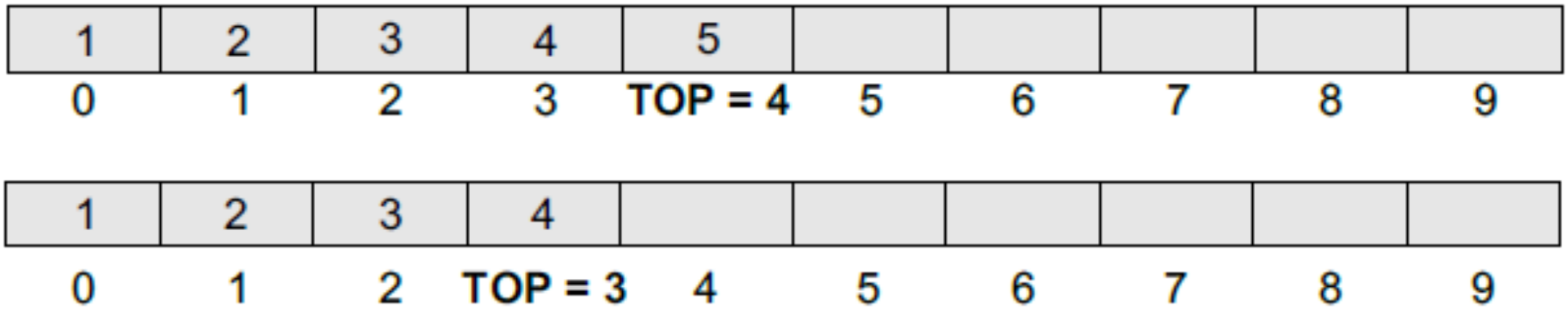
Push(6) causes TOP to be incremented by 1 and 6 stored at STACK[5]

Pop Operation

- The pop operation is used to delete the topmost element from the stack.
- However, before deleting the value, we must first check if TOP=NULL, because it means the stack is empty and no more deletions can be done.
- If an attempt is made to delete a value from a stack that is already empty, an UNDERFLOW message is printed.

Pop Operation

```
Algorithm to POP an element from a stack
Step 1: IF TOP = NULL, then
                PRINT "UNDERFLOW"
                GOTO Step 4
        [END OF IF]
Step 2: SET VAL = STACK[TOP]
Step 3: SET TOP = TOP - 1
Step 4: END
```

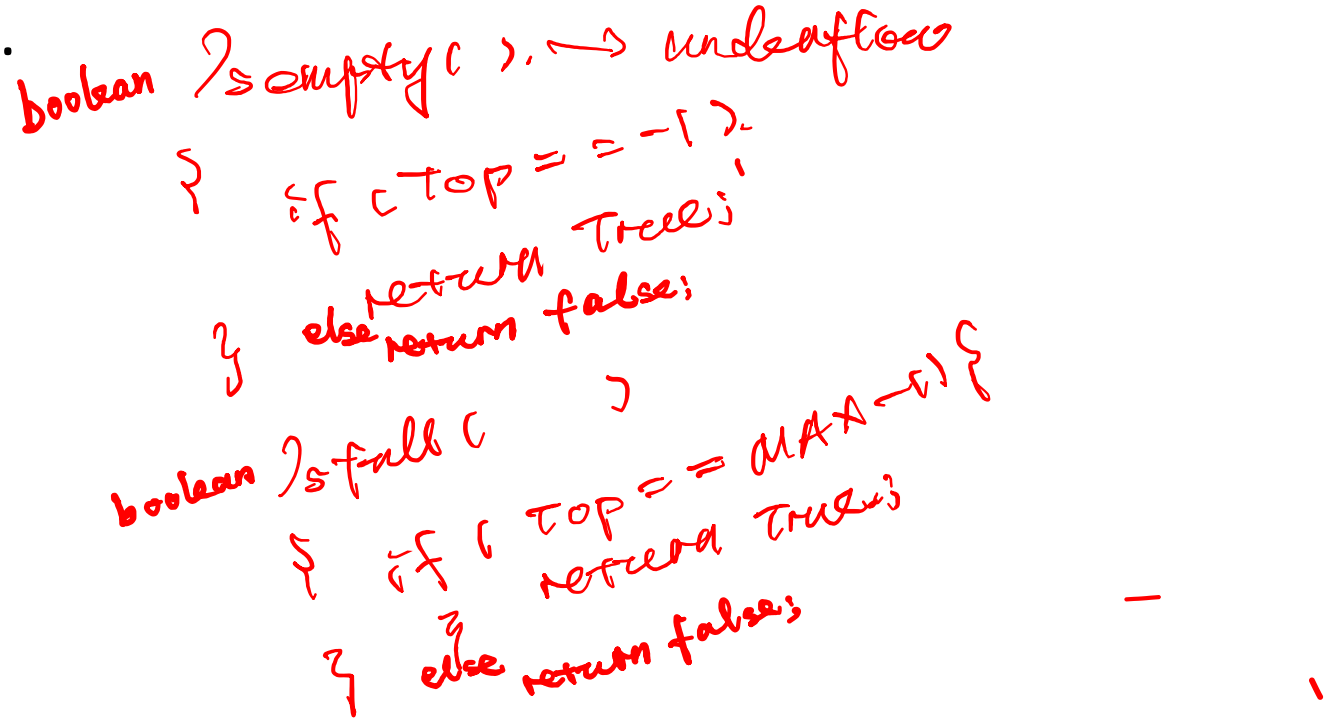


Pop() causes 5 to be stored in VAL and TOP to be decremented by 1

Peek Operation

POP

- Peek is an operation that returns the value of the topmost element of the stack without deleting it from the stack.
- However, the peek operation first checks if the stack is empty.
- If TOP = NULL, then an appropriate message is printed else the value is returned.

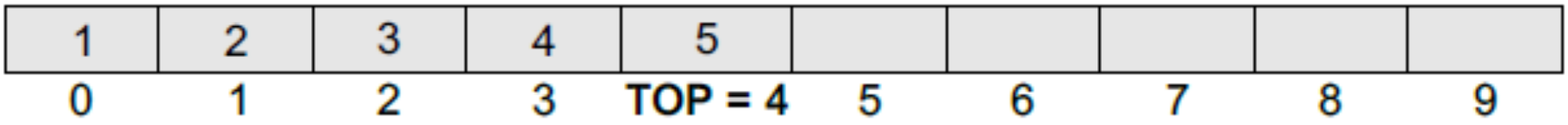


Peek Operation

```
Algorithm for Peek Operation

Step 1: IF TOP = NULL, then
                PRINT "STACK IS EMPTY"
                GOTO Step 3
            [END OF IF]
Step 2: RETURN STACK[TOP]
Step 3: END
```

isEmpty( ).



Peek() returns 5