

## *Stack Mechanism Discussion with Time Complexity*

### *6. Peek Operation*

```
int peek(int stack[])
{
    if (top == -1)
    {
        return -1;
    }

    return stack[top];
}
...
case 3:
cout << "Item at the top is " << peek(stack) << endl;
break;
```

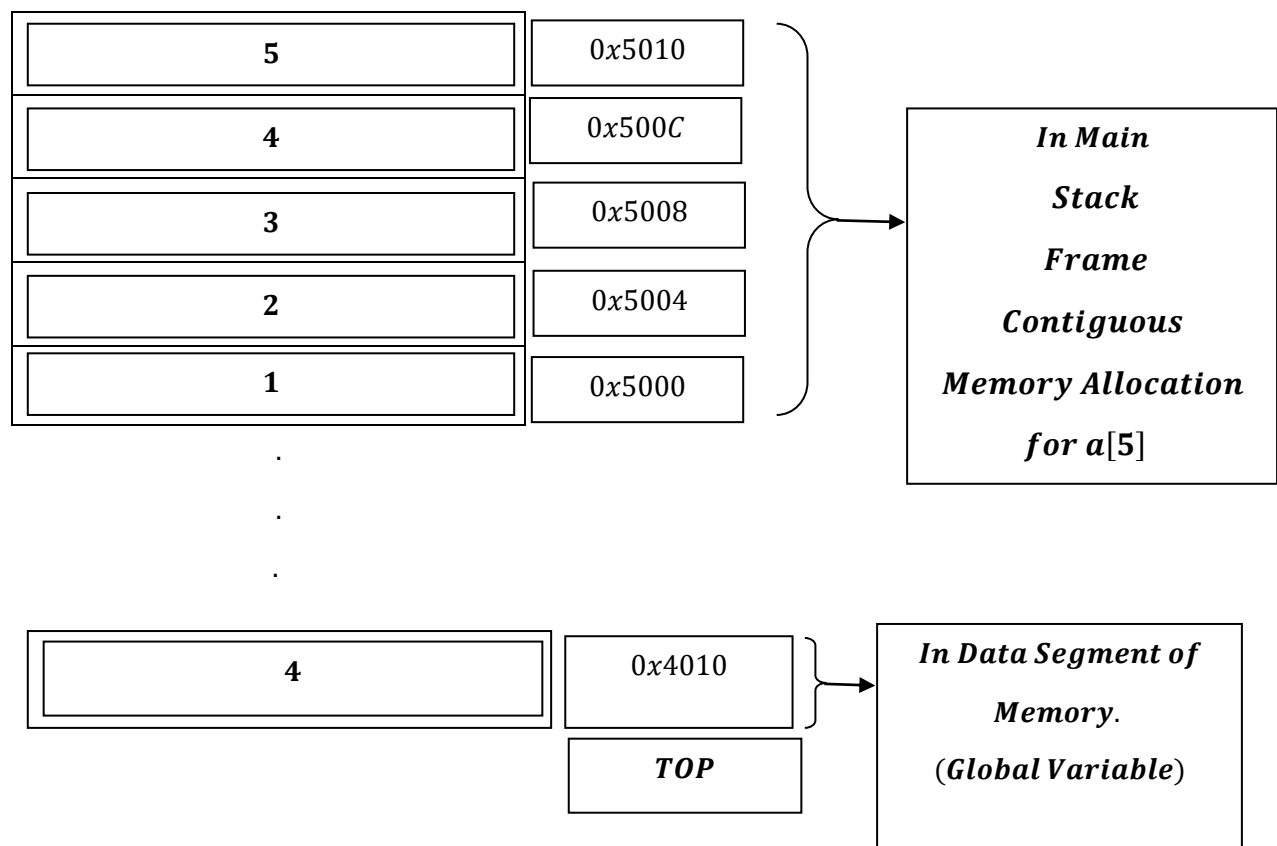
*If ( $top = -1$ ) then:*

*return - 1;*

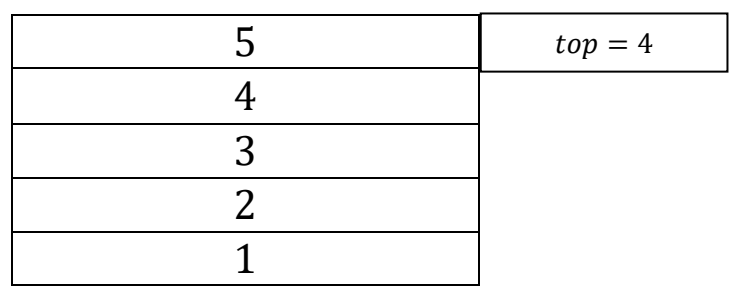
*else:*

*return stack[top];*

***Full Stack***



***This is Physical Demonstration***



***This is Logical Demonstration***

*As now,  $Top = 4$  .*

*$Stack[Top = 4]$ .*

$\Rightarrow Stack + 4$ . [*Stack + 4, represents contiguous memory allocation*]

$\Rightarrow Base\ Address + 4[index] \times 4bytes$ .

$\Rightarrow 0x5000 + 16$ .

$\Rightarrow 0x5000 + 10$ . [ $16_{10} \approx 10_{16}$  ]

$\Rightarrow 0x5010$

*return ``value stored at address: 0x5010`` ; i.e. 5.*

---

### *Time Complexity*

```
int peek(int stack[])
{
    if (top == -1)
    {
        return -1;
    }

    return stack[top];
}
```

→ *Function overhead or stack frame creation when peek() is called takes constant time `c` takes  $O(1)$ .*

→ *if (top = -1) True [Takes constant `c` time :  $O(1)$ ] then:*

→ *return - 1 and exit.  $\left[ \begin{array}{c} \text{Takes constant `c` time:} \\ O(1) \end{array} \right]$*

→ *if (top = -1) False then:*

→ *return stack[top];  $\left[ \begin{array}{c} \text{Takes constant `c` time:} \\ O(1) \end{array} \right]$*

*If true then:*

*Time Complexity =  $O(1) + (O(1) + O(1)) = O(1)$ .*

*If false then:*

*Time Complexity =  $O(1) + (O(1) + O(1)) = O(1)$ .*

---