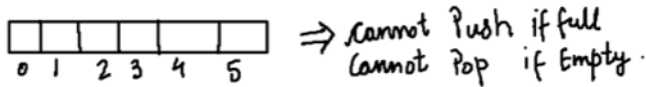


Stack Operations While Implementing Using Arrays



Op 1 \rightarrow Push.

struct stack * sp;

sp \rightarrow size = 8;

sp \rightarrow top = -1;

sp \rightarrow arr = (int *) malloc (sp \rightarrow size * sizeof(int));

\rightarrow Push()

```
if (isFull(sp)) {
    printf("Stack overflow");
}
```

else {

sp \rightarrow top ++;

sp \rightarrow arr[sp \rightarrow top] = val;



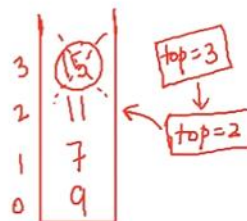
Creating stack

POP Operation

Op 2 \rightarrow Pop :

```
if (isEmpty(sp)) {
    printf("Stack Underflow");
    return -1;
}
```

```
else {
    int val = sp  $\rightarrow$  arr[sp  $\rightarrow$  top];  $\rightarrow$  top = 3
    sp  $\rightarrow$  top = sp  $\rightarrow$  top - 1;  $\rightarrow$  top = 2
    return val
}
```



PEEK Operation

Diagram illustrating the Peek Operation in a Stack using an Array:

The array is represented as an array of 6 cells (indices 0 to 5). The current stack elements are 7, 8, and 12, stored at indices 0, 1, and 2 respectively. The 'Top' pointer is at index 2.

Handwritten code for the peek operation:

```
int peek (struct Stack * sp, int i) {
    if (sp->top - i + 1 < 0)
        printf("Not a valid position");
        return -1;
    else
        return sp->arr[sp->top - i + 1];
}
```

Handwritten notes and diagrams:

- A diagram shows the stack elements 12, 8, and 7 at positions 2, 1, and 0 respectively. The 'top' pointer is at position 2.
- A table showing the mapping between Position (i) and Array Index:

Position (i)	Array Index
1	2
2	1
3	0

- The formula for the array index is given as: $\text{Array Index} = \text{Top} - i + 1$.
- A note indicates $\text{Top} = 2$.
- A handwritten note says: "There is a simple story here, a few lines of code:"

StackTop naam ka function which return the TOPMOST value of the stack

`return sp->arr[sp->top];`

StackBottom returns the LOWERMOST value of the stack

`stackBottom → sp->arr[0];`

Time complexity for both Constant $O(1)$

① is Empty → $O(1)$ ③ Push → $O(1)$ ⑤ Peek → $O(1)$
 ② is full → $O(1)$ ④ Pop → $O(1)$

```
int main(){  
    // struct stack s;  
    // s.size = 80;  
    // s.top = -1;  
    // s.arr = (int *) malloc(s.size * sizeof(int));  
    struct stack *s;  
    s->size = 80;  
    s->top = -1;  
    s->arr = (int *) malloc(s->size * sizeof(int));  
    return 0;  
}
```

Agr object bna rahe to . lagega

Agr pointer bna rahe to -> lagega