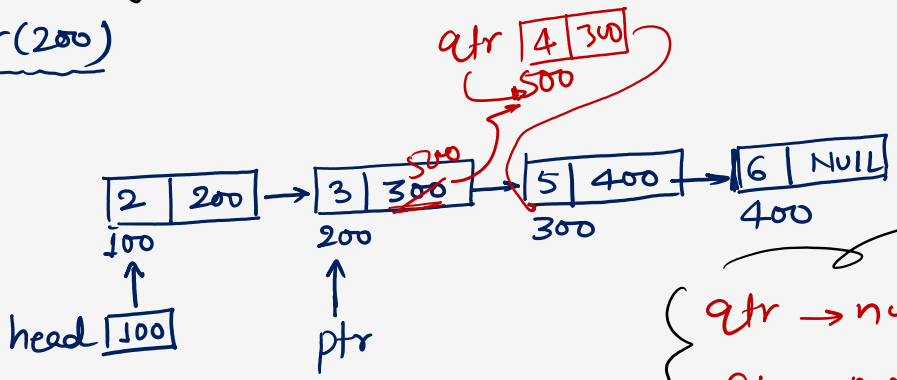
Question:- if ptr(200) given then

i) Insert after ptr(200)

TC= ?



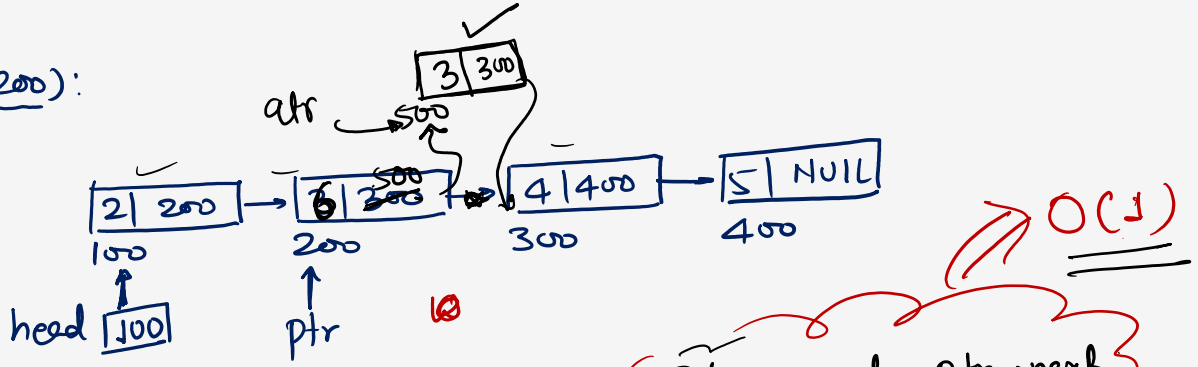

TC=O(1)

```
qtr -> num = 4;
qtr -> next = ptr -> next;
ptr -> next = qtr
```

work for all node

## (ii) Insert before ptr(200):

TC = ?



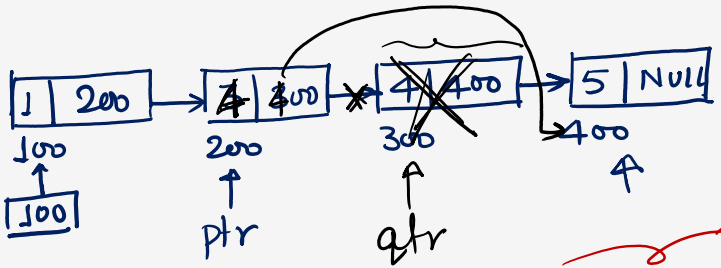

O(n)

qtr → next = ptr → next;
qtr → num = ptr → num
ptr → next = qtr
ptr → num = item

O(1)

Work for all node.

iii) Delete at ptr (200):-




TC = ?

Best Known Algo

This Concept is not work for last node.

BC = O(1)

WC = O(n)

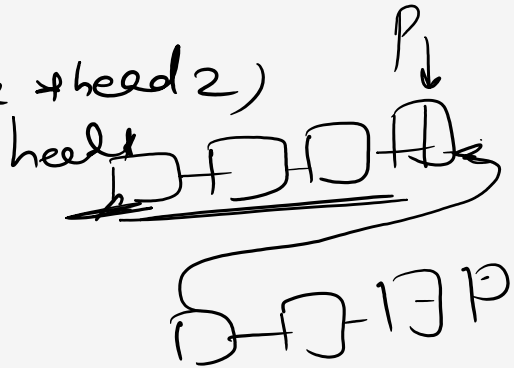
O(1)

```
qtr = ptr->next;
ptr->num = qtr->num;
ptr->next = qtr->next;
free(qtr);
```

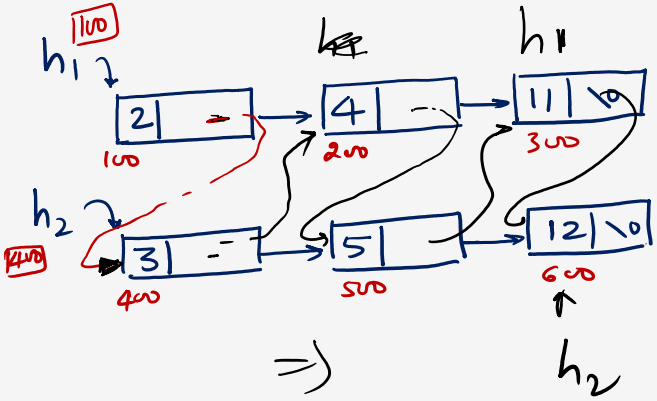
NOTE: In the SLL, Pointer P is given Pointing to some node, then deleting that node, what is the TC = O(n)

## Concatenation of linked list:-

```
node * Concatenate ( node *head1, node *head2)
{ node *p
  if (head1 == NULL)
        return (head2);
  if (head2 == NULL)
        return(head1);
  p= head1;
  while( p->next1 =NULL)
        p=p-> next;
   p->next = head2;
  return(head1);
}
```



## Merge two Sorted list:-

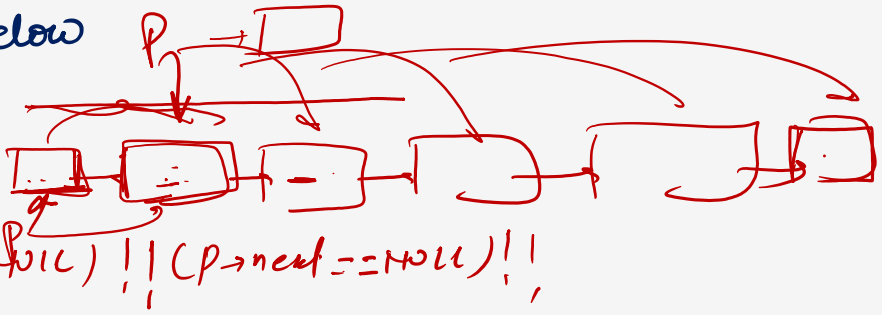



$\Rightarrow$

Linked list :-

```
node * merge (node *h1, node *h2)
{
  if (!h1)
     return h2;
  if(!h2)
     return h1;
  if(h1->data < h2->data)
    { h1->next = merge(h1->next, h2);
      return h1;
    }
  else { h2->next = merge(h1, h2->next);
         return h2;
       }
}
```

**Question:-** Consider the function f defined below




```
Struct item
{ int data;
  Struct item *next;
};
int f (Struct item *P)
{ return ((P==NULL) || (P→next==NULL) || (P→data<=P→next→data && f(P→next)));
}
```

1 && f(P→next)

for a given linked list P, the function f returns 1 iff

50%.

a) The list is empty or has exactly one element

b) The elements in the list are sorted in non decreasing order of data values

Question: Let P be a singly linked list. Let Q be the pointer to an intermediate node x in the list what is the worst case time complexity of the best known Algo. to delete the node x from the list

a) $O(n)$

b) $O(\log^2 n)$

c) $O(\log n)$

d) $O(1)$

Question: What are the time complexity of finding 8th element from begning and 8th element from end in singly linked list

Let n be the no of nodes in linked list you may assume that n>8

a) O(1), O(n)

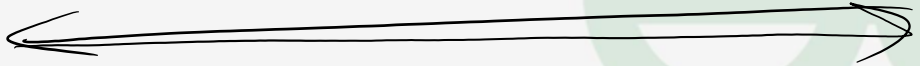
b) O(1), O(1)

c) O(n), O(1)

d) O(n), O(n)

O(1), O(n)

O(1) → O(n)



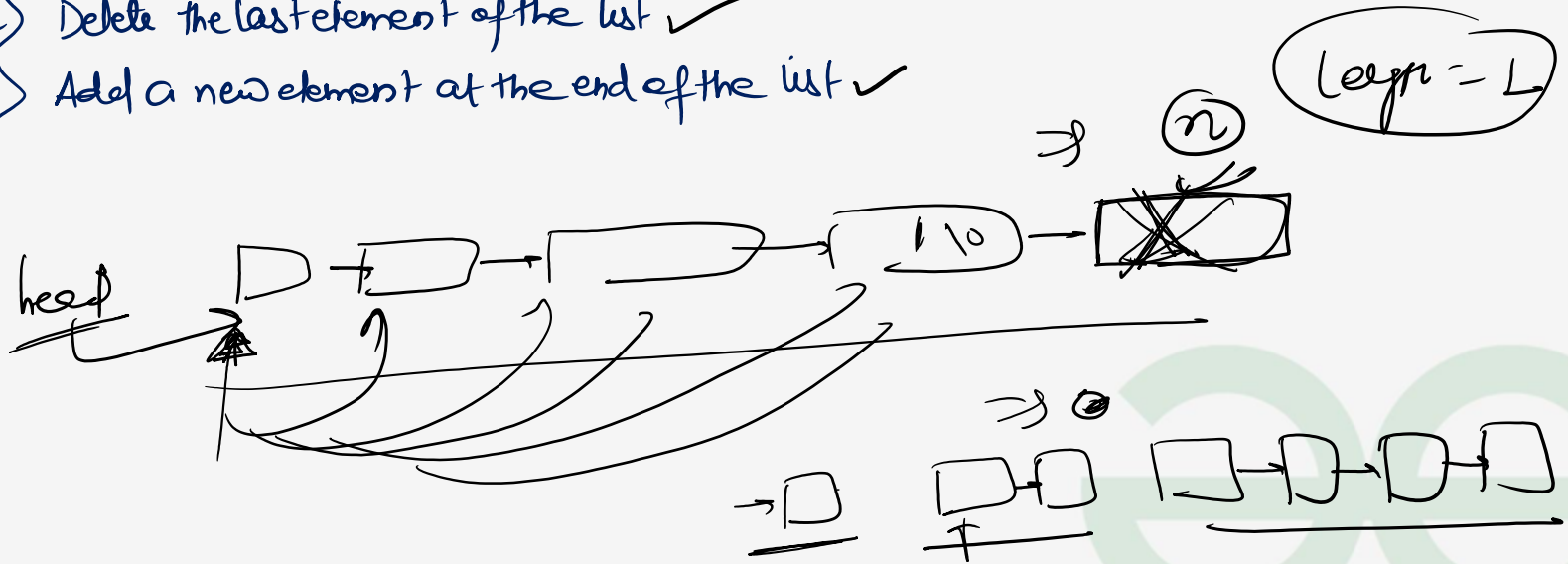
Question: Which of the following operation are dependent on the Length of the Linked list

a) Delete the first element X

b) Insert a new element as a first element X

c) Delete the last element of the list ✓

d) Add a new element at the end of the list ✓



## Linear Search :-

```
int Search(int x)
{ node *p;
  p = head;
  while(p != NULL)
  {
    if(p->data == x)
      return 1;
    p = p->next;
  }
  return(0);
}
```

Binary Search :-

$T(n)$

```
node * LLBS (node *S, x)
{ if (S -> next == NULL)
  { if (S -> data == x)
      return S;
    else return NULL;
  }
  else
  { m = midLL(S);
    if (m -> data == x)
      return m;
    else if (m -> data < x)
      LLBS(m -> next, x);
    else
    { m -> next = NULL;
      LLBS(S, x);
    }
  }
}
```


O(1)

Recurrence Relation:-

$$T(n) = \begin{cases} 1, & \text{if } n=1 \\ T(n/2) + n, & \text{if } n>1 \end{cases}$$

$T(n) = T(n/2) + n$

$T \Rightarrow O(n) \Rightarrow$

mid LL(S) → O(n)

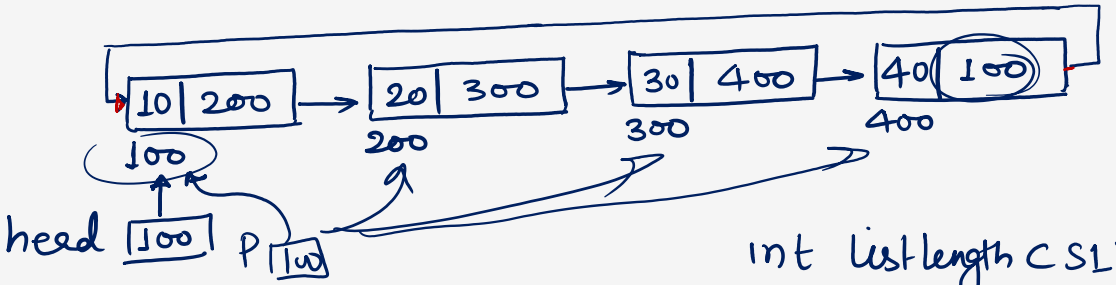
if (m -> data == x) → O(1)

LLBS(m -> next, x) → $T(n/2)$

LLBS(S, x) → $T(n/2)$

# Circular Single linked list (CSLL or SCLL) :-

```
P=head;
while( P->next != HULL)
    P=P->next.
```

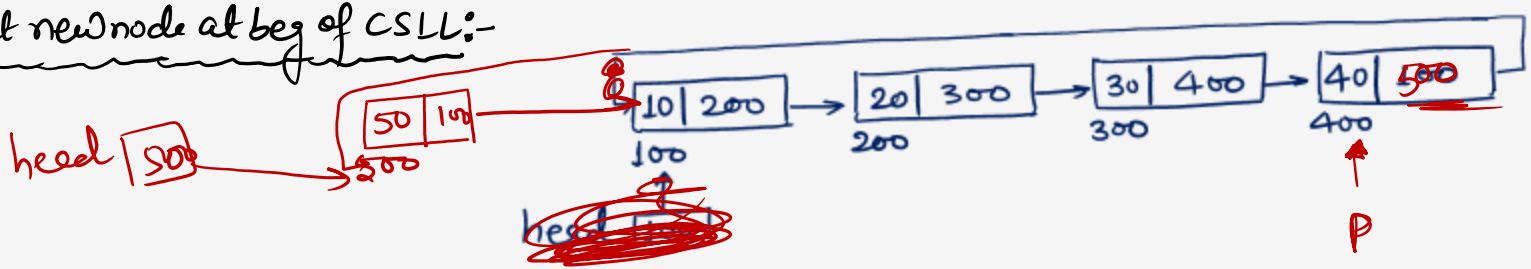



## Count No. of nodes in the given Circular linked list:-

Count = 1 2 3 4

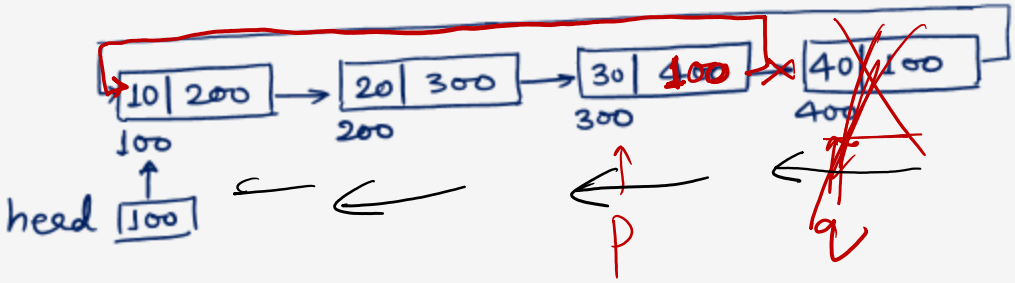
```
int listlength CSLL (node *head)
{ if(head==NULL) return0;
  else
  { int Count=1;
    node *P =head;
    while(P->next != head)
    { P=P->next;
      Count++;
    }
    return Count;
}
```

Insert newnode at beg of CSLL:-




O(n)

Delete last node of a CSLL:-




→ O(n)

| CSLL | Begin | End |
|---|---|---|
| Insert | O(n) | O(n) |
| Delete | O(n) | O(n) |

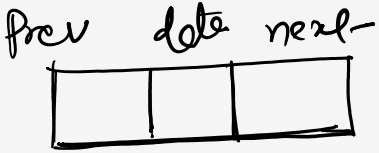
p → next = q → next

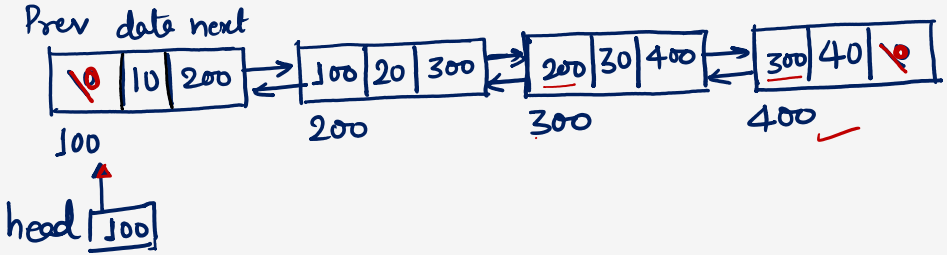
free(q)

p = NULL

q = NULL

# Double linked list (DLL):-



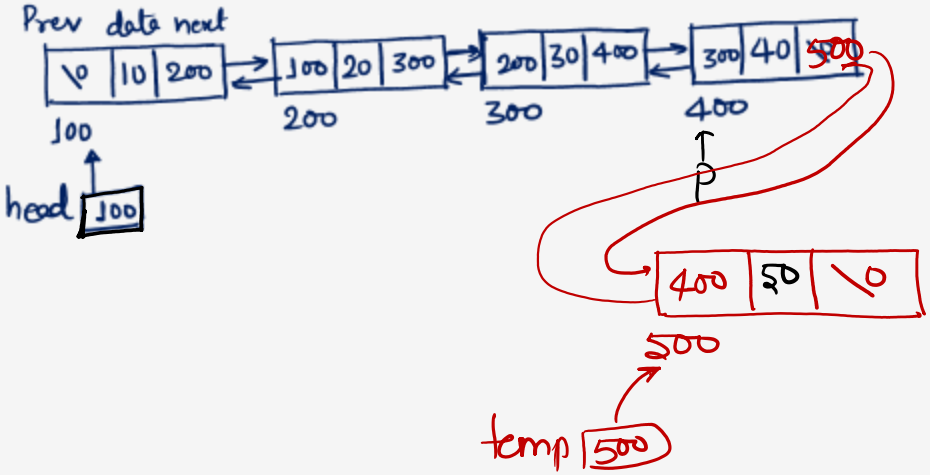




```
Struct DLLnode
{ Struct DLLnode *Prev;
  int data;
  Struct DLLnode *next;
};
Typedef Struct DLLnode node;
```

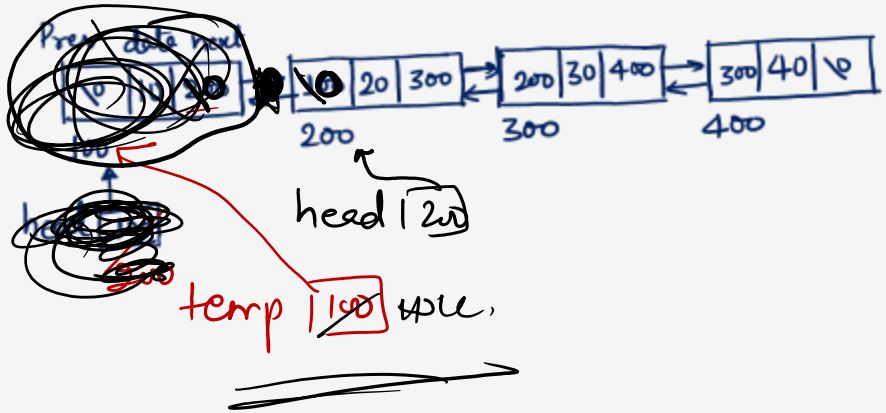
NOTE:-

1) In the DLL every node Contains two Pointers Pointing to Previous node & next node
2) We can traverse in both the directions
3) Insertion and deletion will take more time because of more Pointers updation

Insert new node at end of DLL:-



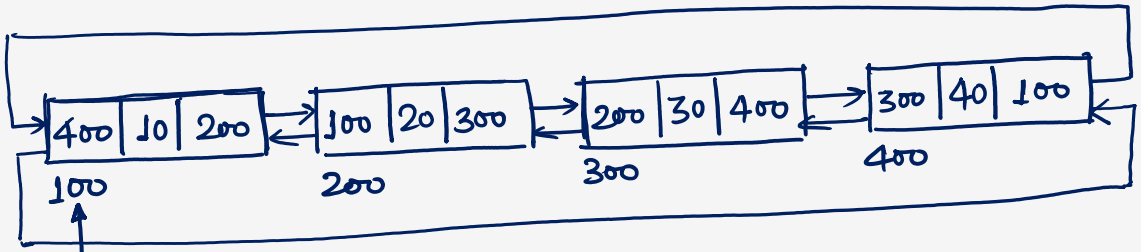
## Delete at begin in DLL:-




| DLL | Begin | end |
|---|---|---|
| Insert | O(1) | O(n) |
| Delete | O(1) | O(n) |

## Circular Double linked list (CDLL):-

In the circular Double linked list the first node Previous Pointing to last node and last node next will point to first node




Last node Insertion:

| DLL | Begin | Last |
|---|---|---|
| Insert | O(1) | O(1) |
| Last | O(1) | O(1) |

Question:- The concatenation of two list is to be performed in O(1) time which of the following implementation of a list should be used

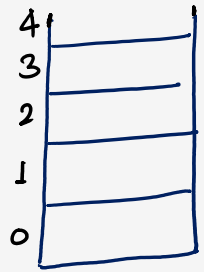
a) singly linked list O(n) ——— O(n)

b) doubly linked list O(n)

c) Circular doubly list O(1)

d) Array implementation of list

STACK:- Stack is a data structure used to store the data elements with the restriction of LIFO (FILO), The elements are inserted and deleted from the same end, called "Top",



STACK

Top = 4 3 2 1 0 -1

Initially: Top = -1 0 1 2 3 4

```
#define N 5
Char STACK[N];
```

Q-1) A program attempts to generate as many Permutations as Possible of the String "abcd" by Pushing the characters a, b, c, d in the order onto a STACK, but it may Pop off the Top character at any time. Which one of the following Strings can not be generated using This Program?

a) abcd  b) dcba  c) cbad  d) cabd  Ans

Q-2) Which of the following Permutation can be generated in the output (in the same order) using a STACK assuming that input is the sequence 1,2,3,4,5 in that order?

a) 3,4,5,1,2 X

b) 1,5,2,3,4 X

c) 3,4,5,2,1

d) 5,4,3,1,2 X

3 4 5 2 1

## Applications of STACK:-

1) Recursive function calls
2) Balancing Parenthesis
3) Checking html and xml tag balancing
4) Page visited in browser
5) Infix to Postfix Conversion
6) Postfix evaluation
7) fibanacci series
8) Tower of Hanoi
9) Undo & Redo in text editor etc.

ADT of STACK: What operations we can Perform on STACK

1) Push(x):- An element to insert

The above operation insert an element in the STACK with the help of top

2) POP():- Delete an element

The above operation delete an element from the STACK with the help of Top

3) isEmpty(): Return TRUE when the STACK is empty

4) isfull(): Return TRUE when the STACK is full

## Implementation of STACK:-

### Using Array :-

```
#define N 5
int STACK[N];
```

```
int S[N];
```

⇒

```
Algo: Push(S, N, Top, x)
{
    if(Top == N-1)
    { pf("STACK is full");
      return
    }
    else
    {
        Top++;
        S[Top] = x;
    }
}
```

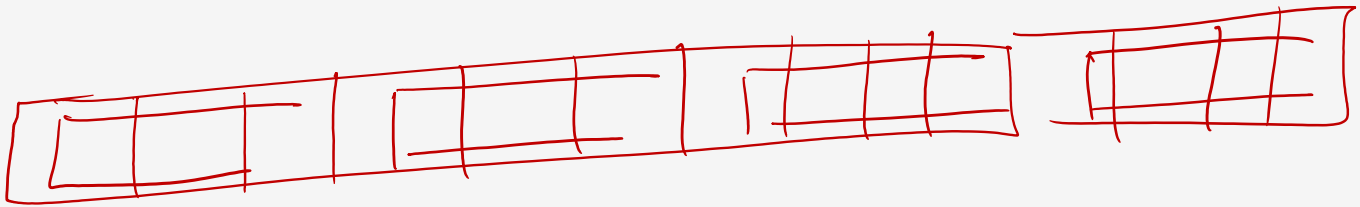
STACK, SIZE, TOP, element

TC = O(1)

TC =

```
int Pop(S, N, Top)
{
    if(Top == -1)
    { pf("STACK is empty");
      return -1;
    }
    else
    { y = S[TOP];
      top--;
    }
    return y;
}
```

TC = O(1) ⇒



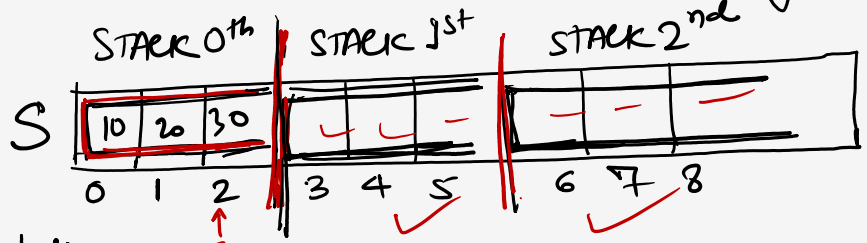
## Implementation of multiple STACKS in Single Array:

```
int S[9];
```

Let Size of Array (N) = 9

No. of STACKS (M) = 3

Size of each STACK = $\frac{N}{m} = \frac{9}{3} = 3$




Initially

| | 0th STACK | 1st STACK | 2nd STACK | ---- | eth STACK |
|---|---|---|---|---|---|
| Empty | $T_0 = -1$ | $T_1 = 2$ | $T_2 = 5$ | ---- | $e(\frac{N}{M}) - 1$ |
| Full: | $T_0 = 2$ | $T_1 = 5$ | $T_2 = 8$ | ---- | $(e+1)(\frac{N}{M}) - 1$ |

$$T_e = = e(N/M) - 1$$

$$T_e = = (e+1)(\frac{N}{m}) - 1$$

Algo: Push(S, N, M, T_i, i, x)

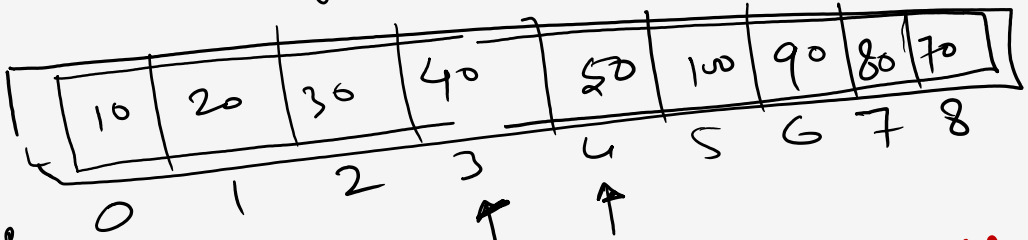
```
{
  if( T_i == (i+1) N/M - 1 )
    { pf("STACK is full");
      exit(1);
    }
  else
    { T_i++;
      S[T_i] = x;
    }
}
```

Algo: Pop(S, N, M, T_i, i)

```
{
  if( T_i == i*N/M - 1 )
    { pf("STACK is empty");
      exit(1);
    }
  else
    { y = S[T_i];
      T_i--;
      return T_i;
    }
}
```

# Efficient way of Implementing multiple STACK:-

2 STACK




Initially TOP

$T_R = 9\ (N)$

8 7 6 5 4

Initially TOP

$T_L = -1$ 0 1 2 3 4 5 $T_L$ $T_R$

Overflow Condition

Imp

$T_L = T_R - 1$

$T_R = T_L + 1$

$T_R - T_L = 1$

# Thank You !