UNIT - II : Stack

Friday, May 2, 2025 6:27 PM

→ Linear data structures

↳ we follow a sequence

Rule:- Last in First Out → what is the element inserted at ~~first~~ Last- that will be deleted first-

Example:- girl → wear abangle

Operations:-

1) push → insert a new element on to pof stack
2) pop → delete an element from top stack
3) display → is to print the values
4) isFull → to test whether stack is full or not

Implementation Methods:-

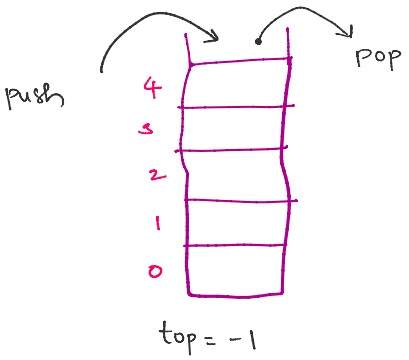
1) using Array (II unit)
2) using linked list (III unit)

* Implementation of stack Using Array:-

Requirements:- Array[5] → stack[5] → name of stack

MAX = 5 (maximum size of stack)

top = -1 (no elements in the stack)

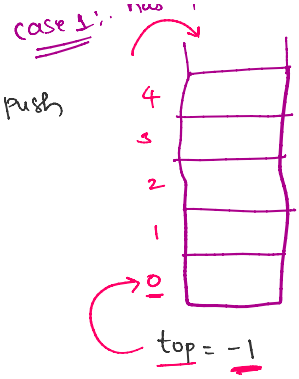


push:- insert new element on top of the stack

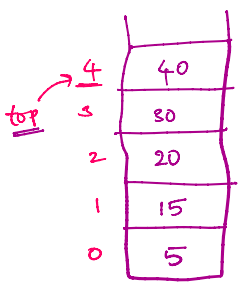
case 1:- has space to insert

case 2:- stack is full

Case 1:- Full



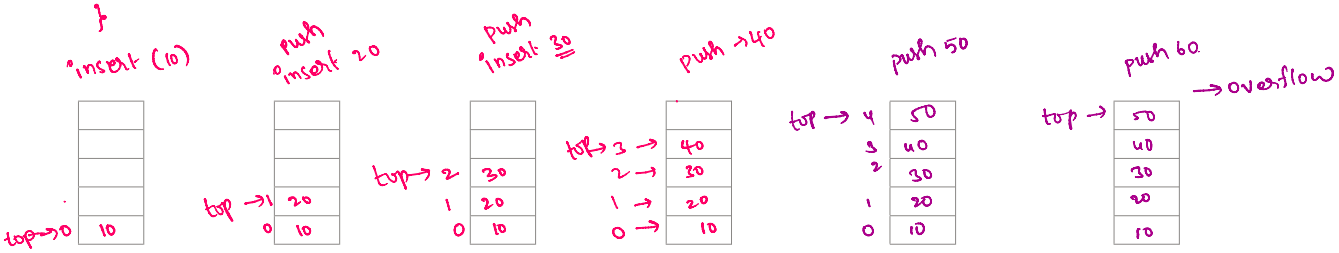
Case 2:- Stack is full



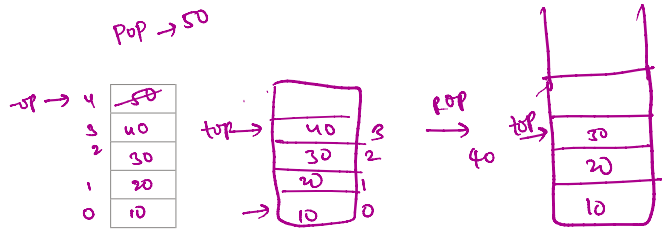
MAX - 1
5 - 1 = 4 ← top

```
void push(int *item)
{
    if(top == MAX-1)
    {
        print("stack is full → Overflow");
    }
    else
    {
        top = top+1;
        stack[top] = *item;
    }
}
```

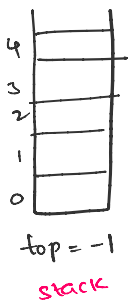
step 1:- test stack is full or not
if full → print "overflow"
if not full →
top is incremented by '1'
insert element at 'top' position



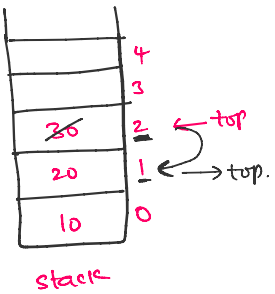
Pop:- Delete an element from top of the stack



Case 1:-
empty



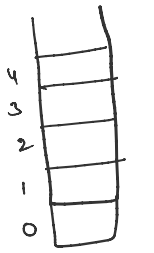
Case 2:-



```
void pop()
{
   if(top == -1)
   {
      print("stack is empty - underflow");
   }
   else
   {
      print(" deleted element is %d", stack[top]);      → 30
      top = top - 1;
   }
}
```

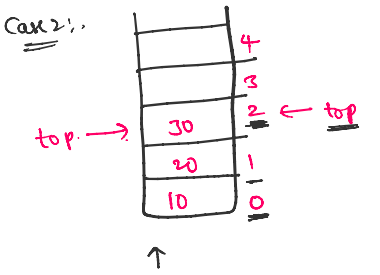
display → it prints what the elements are in stack : from " top to 0 index "

Case 1:



top = -1

Case 2:



i= i= i=

2nd 1st 0th

print → 30 , 20 , 10

```
void display()
{
   if ( top == -1)
   {
      print("Stack is Empty -No elements to display");
   }
   else {
     for( int i=top ; i >=0; i--)
      {
         printf(" %d", stack[i]);
      }
}
```

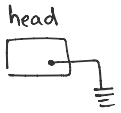
isFull() → it test whether stack is full or not

```
void isFull()
{
   if ( top == MAX-1)
   {
      print(" stack is Full");
   }
   else
   {
      print("Stack is not full");
   }
```

}

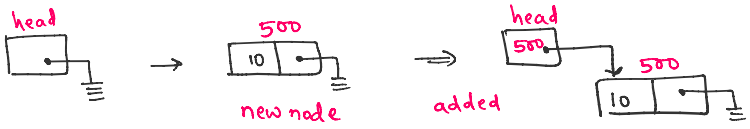
}

Implementation of stack Using Linked List:-

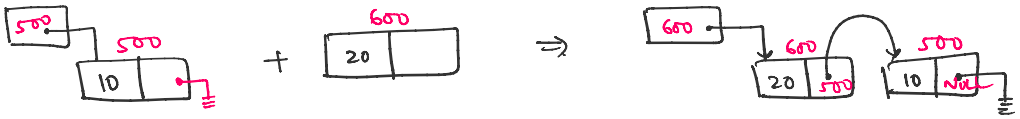
Step 1:- 

→ list is empty

Step 2:- push (10)

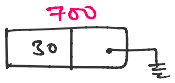


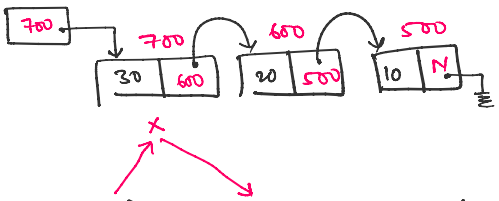
Step 3:- push (20) →



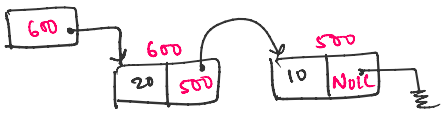
inserting node at beginning of single linked list

Step 4:- push (30) :

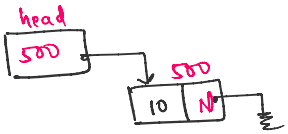




Step 5:- pop() → deleted element is : 30



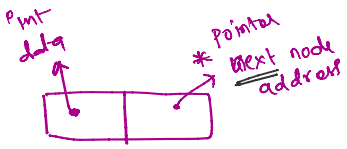
Step 6:- pop() → deleted element is : 20



delete an element at beginning of single list

Requirements:

i) node structure :-



```
Struct Node
{
    int data;
    Struct Node  * next;
}
```

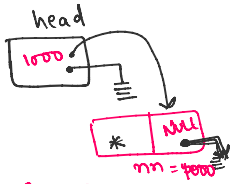
(ii) Create a new node:-

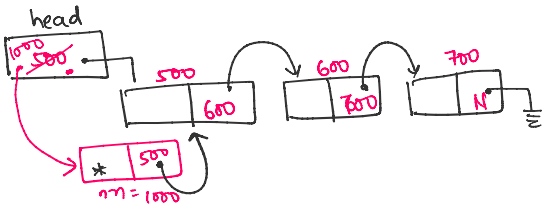
```
Struct Node  * nn = (Struct Node *) malloc (sizeof (Struct Node));
```

(iii) Create a head

```
struct Node  *head = NULL;
```

push():-

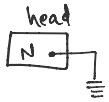




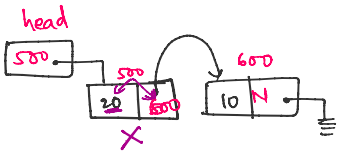

```
void push(int item)
{
    Struct Node  *nn = (Struct Node *) malloc (size of (Struct Node *));
        nn -> data = item;
    if( head == NULL)
    {
        nn -> next = NULL;
        head = nn;
    }
    else
    {
        nn -> next = head;
        head = nn;
    }
}
```

Pop:-

Case 1:



Case 2:

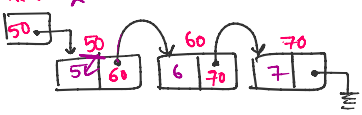



```
Void Pop()
{
    if (head == NULL)
    {
        print(" Stack is Underflow");
    }
    else
    {
        print(" deleted element %d", head -> data);
```

```
    {
        print(" deleted element %d", head->data);
        head. head->next;
    }
}
```

Case 1:
head

Case 2:
head




```
void display ( )  → printing stack elements
{
    if (head == NULL)
    {
        printf(" stack is empty");
    }
    else
    {
        Struct Node *tl=head
        while (tl!=NULL)
        {
            printf(" %d", tl->data);
            tl = tl->next;
        }
    }
}
```