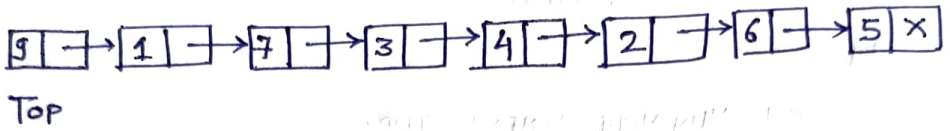
LINKED REPRESENTATION OF STACK.

In a linked stack, every node has two part → One that stores data and another that stores the address of the next node.

The START POINTER of the linked list is used as TOP.

All the insertion and deletion are done at the node pointed by Top.

If TOP == NULL, then it indicates that stack is empty.



LINKED STACK.

Operations on a linked list

A linked list supports all the three stack operations, that is push, pop and peek.

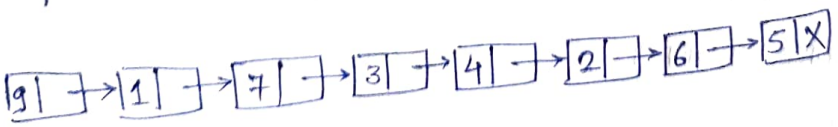
(a) Push Operation

New element added at the topmost position of the stack.

To insert new element with value 9, first check if TOP == NULL.
In this case, we allocate memory for a new node, store value in its DATA PART and NULL in its next part.

However if TOP != NULL, then we insert a new node at the beginning of the linked stack and name this new node as TOP.

Thus update stack;



PUSH

Step1: Allocate memory for the new node and name it as NEWNODE.

Step2: SET NEWNODE → DATA = VAL;

Step3: IF TOP = NULL

SET NEWNODE → NEXT = NULL;

SET TOP = NEW-NODE

ELSE

SET NEW-NODE → NEXT = TOP;

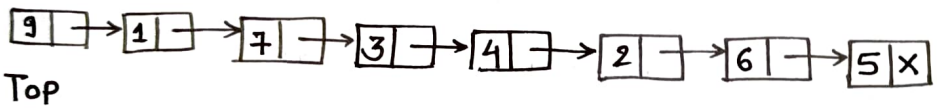
SET TOP = NEW-NODE

[END OF IF]

Step 4: END

Pop Operation.

1).



2).

Step1: IF TOP = NULL

PRINT "UNDERFLOW"

Goto Step5

[END OF IF]

Step2: SET PTR = TOP;

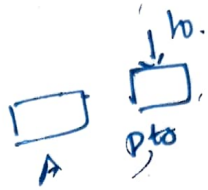
Step3: SET TOP = TOP → NEXT

Step4: FREE PTR

Step5: END

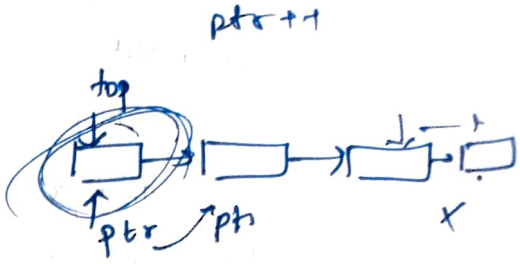
```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <stdlib.h>

Struct stack
  {
    int data;
    Struct Stack *next;
  };
Struct stack *top=NULL;
Struct stack *push (struct stack*, int);
Struct stack * pop (Struct Stack *);
int peek (Struct Stack*);

int main (int argc, char *argv[]
  {
   int val, option;
   do
   {
      ---------- printf
      ----------  "
      ----------  "
      ----------  "
     switch (option)
      {
      case 1: printf enter the number to be pushed on stack:");
              scanf ("%d",&val);
              top= push (top,val);
              break;

       case 2:
              top= pop(top);
              break;

       case 3:
              val= peek(top);
              if (val!=-1)
              printf("\n val at the top of stack is: %d", val);
              else
              printf ("\n STACK IS EMPTY");
              break;
```

```
struct stack *push (struct stack *top, int val)
{
  struct stack *ptr;
  ptr = (struct stack *) malloc (sizeof (struct stack));
  ptr->data = val;
  if (top == NULL)
  {
    ptr->next = NULL;
    top = ptr;
  }
  else
  {
    ptr->next = top;
    top = ptr;
  }
  return top;
}

struct stack *pop (struct stack * top)
{
  struct stack *ptr;
  ptr = top;
  if (top == NULL)
  printf ("\n STACK UNDERFLOW");
  else
  {
    top = top->next;
    printf ("\n The value being deleted is : %d", ptr->data);
    free (ptr);
  }
  return top;
}
```



ptr = top

```
struct stack *display (struct stack *top)
{
    struct stack *ptr;
    ptr = top;
    if (top == NULL)
    printf ("\n STACK IS EMPTY");
    else
    {
    while (ptr != NULL)
      {
       printf ("\n%d", ptr->data);
         ptr = ptr->next;
      }
}
return top;
```



ptr->next != NU

```
int peek (struct stack *top)
{
  if (top == NULL)
   return -1;
  else
   return top->data;
}
```