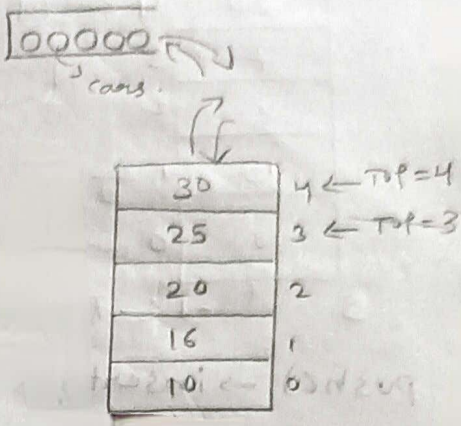
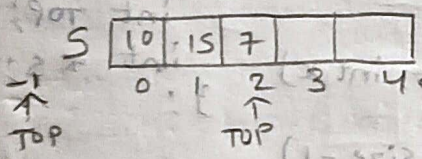
Stack -> set of elements

LIFO



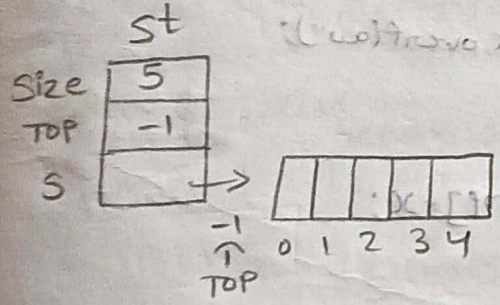
Stack using array

Size=5



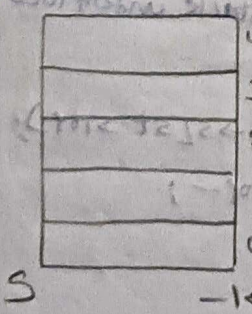
O(1)

```
struct stack
{
    int size;
    int TOP;
    int *s;
};
```




```
int main()
{
    struct stack st;
    printf("Enter size of stack");
    scanf("%d", &st.size);
    st.s = new int [st.size];
    st.TOP = -1;
}
```

Size=5



Empty

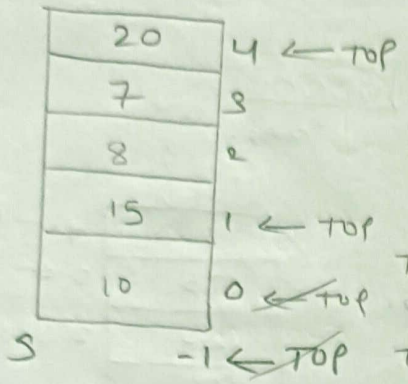
if (TOP == -1)

FULL

if (TOP == size-1)

Size=5

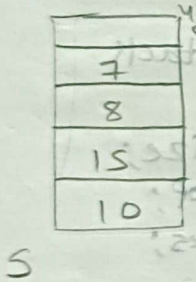
push



push() → inserting an ele in stack.

* ⇒ So before insertion check whether the stack is full (or) not if it is full, we cannot insert it. if it is not full, I can increment top pointer.

Size=5

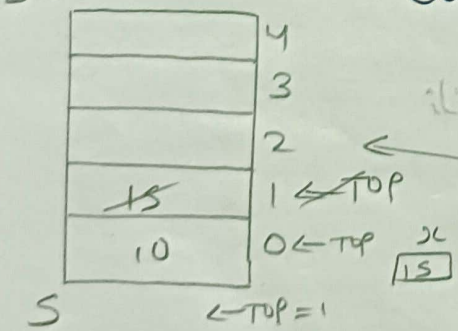



```
struct stack
{
  int size;
  int top;
  int *s;
};
```

```
void push(stack *st, int x)
{
  if(st->top==st->size-1)
    printf("stack overflow");
  else
  {
    st->top++;
    st->s[st->top]=x;
  }
}
```

Size=5

pop

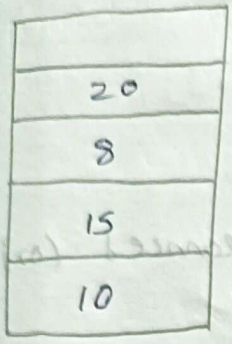



```
struct stack
{ int size;
  int top;
  int *s;
};
```

```
int pop(stack *st)
{
  int x=-1;
  if(st->top==-1)
    printf("stack underflow");
  else
  {
    x=st->s[st->top];
    st->top--;
  }
  return x;
}
```

Peek

Size = 5

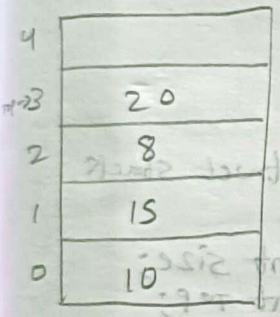



```
Struct stack
{ int size;
  int top;
  int *s;
};
```

pos	index = top - pos + 1
1	3 = 3 - 1 + 1
2	2 = 3 - 2 + 1
3	1 = 3 - 3 + 1
4	0 = 3 - 4 + 1

```
int peek(stack st, int pos)
{ int x = -1;
  if (st.top - pos + 1 < 0)
    printf("Invalid position");
  else
    x = st.s[st.top - pos + 1];
  return x;
}
```

Size = 5



stack top

```
int stackTop(stack st)
{ if (st.top == -1)
    return -1;
  else
    return st.s[st.top];
}
```

```
struct stack
{ int size;
  int top;
  int *s;
};
```

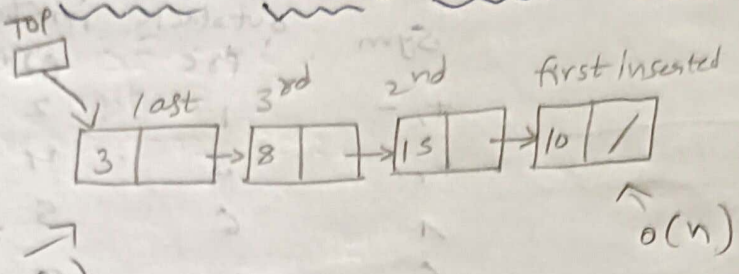
isEmpty

```
int isEmpty(stack st)
{ if (st.top == -1)
    return 1;
  else
    return 0;
}
```

isFull

```
int isFull(stack st)
{
  if (st.top == st.size-1)
    return 1;
  else
    return 0;
}
```

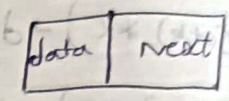
Stack using Linked List



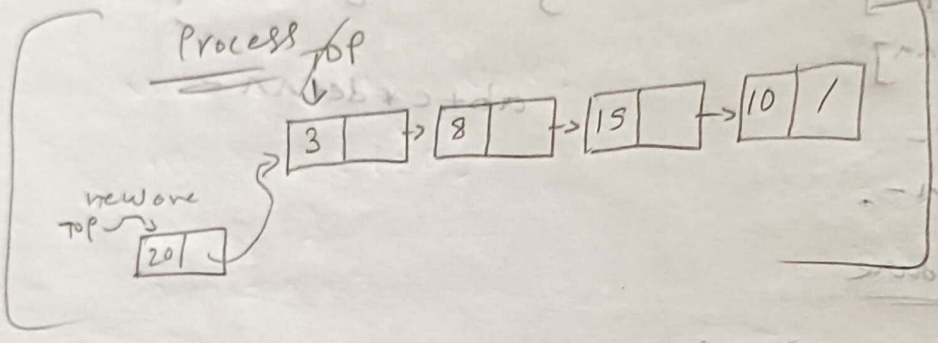
best → O(1)

O(n)

Node




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



Empty

if (TOP == NULL)

FULL

struct Node *t = new Node;

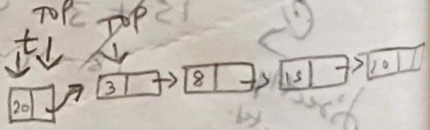
if (t == NULL)

when heap full then t == NULL

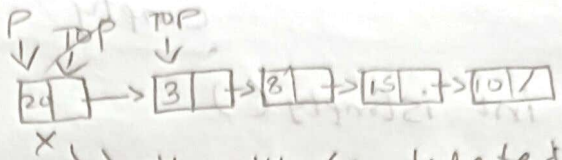
Stack opeans using LL

push

```
void push(int x)
{
    struct Node *t = new Node;
    if (t == NULL)
      printf("stack overflow");
    else
    { t->data = x;
      t->next = TOP;
       TOP = t;
    }
}
```



Pop




```
int pop( )
{ struct Node *P;
  int x=-1;
  if(TOP==NULL)
    printf("stack is Emety");
  else
  {
      P=TOP;
      TOP=TOP->next;
      x=P->data;
  }   free(P);
  return x;
}
```

peek

-> which takes position & gives the value at the position.

-> if position->1 takes TOP value.

```
int Peek(int pos)
{  int x=-1;-> no need
   Node *P=TOP;
   int i;
  for(i=0; P!=NULL && i<pos-1; i++)
  {
      P=P->next;
  }
  if(P!=NULL)
      return P->data;
  else
      return -1;
}
```

StackTop

```
int stackTop( )
{
  if(TOP)
    return TOP->data;
  return -1;
}
```

Empty

```
int isEmpty()
{
return top?0:1;
}
```

full

```
int isFull()
{
struct Node *t = new Node;
    int r=t?1:0;
    free(t);
    return r;
}
```

Code using LL stack pdf