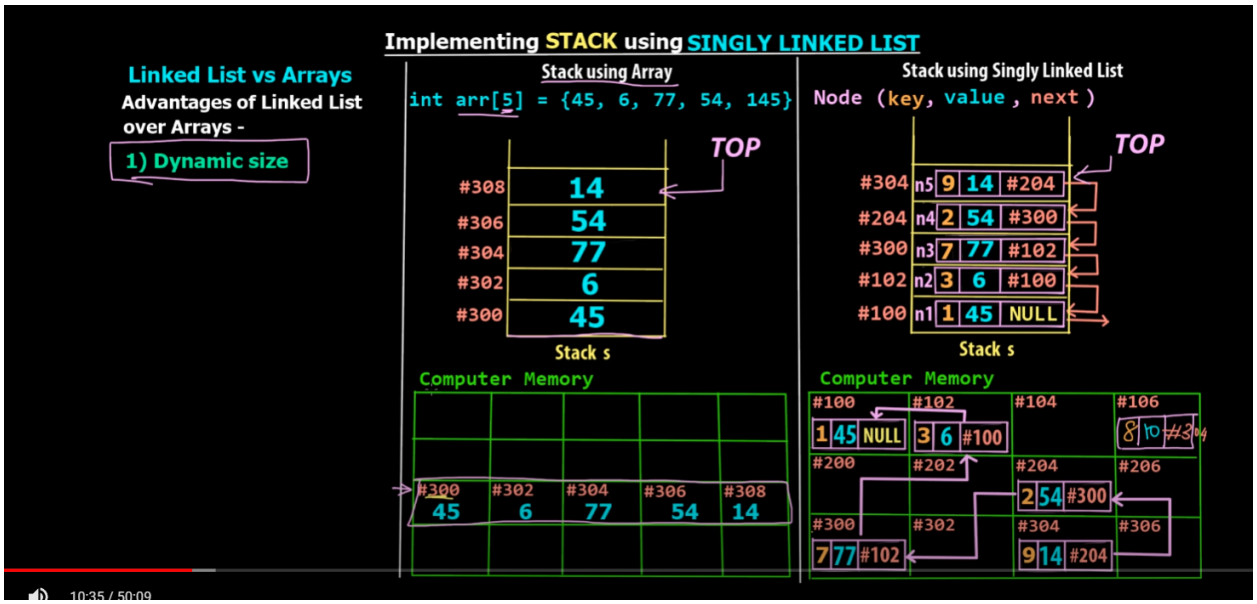
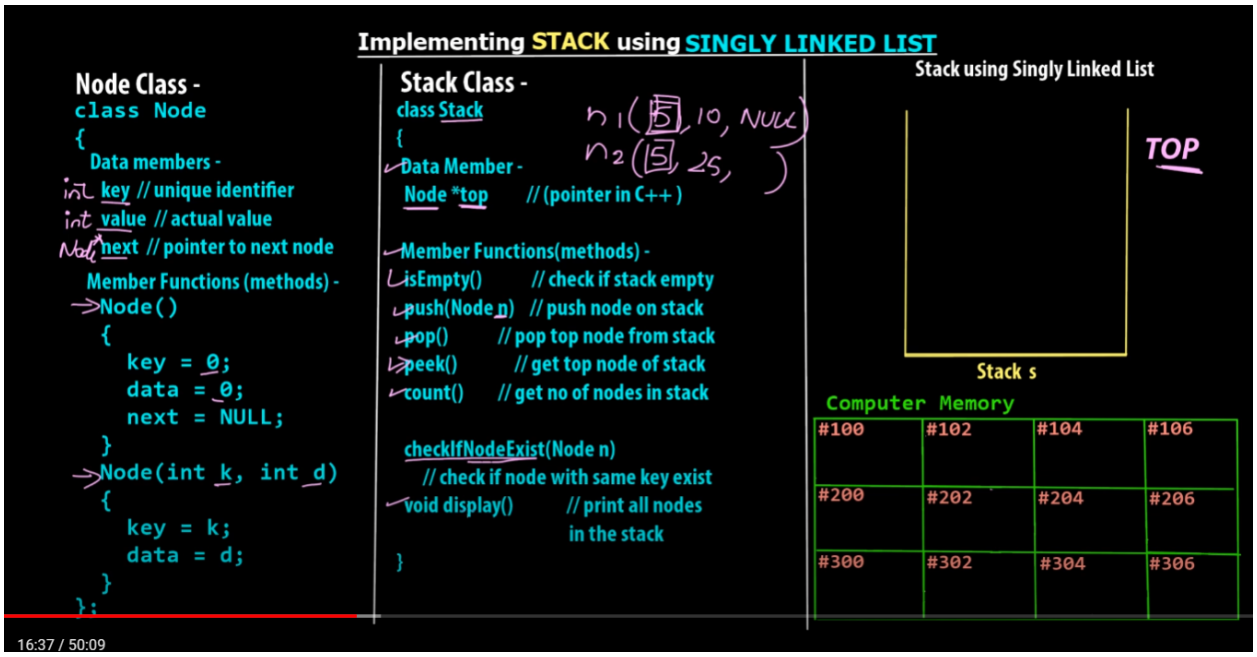

Implementing STACK using SINGLY LINKED LIST
Linked List vs Arrays
Advantages of Linked List over Arrays -
1) Dynamic size
Stack using Array
int arr[5] = {45, 6, 77, 54, 145}
TOP
#308 14
#306 54
#304 77
#302 6
#300 45
Stack s
Computer Memory
#300 45
#302 6
#304 77
#306 54
#308 14
Stack using Singly Linked List
Node (key, value, next)
TOP
#304 n5 9 14 #204
#204 n4 2 54 #300
#300 n3 7 77 #102
#102 n2 3 6 #100
#100 n1 1 45 NULL
Stack s
Computer Memory
#100 1 45 NULL
#102 3 6 #100
#104
#106
#200
#202
#204 2 54 #300
#206
#300 7 77 #102
#302
#304 9 14 #204
#306
10:35 / 50:09


Implementing STACK using SINGLY LINKED LIST
Node Class -
class Node
{
Data members -
int key // unique identifier
int value // actual value
Node next // pointer to next node
Member Functions (methods) -
Node()
{
key = 0;
data = 0;
next = NULL;
}
Node(int k, int d)
{
key = k;
data = d;
}
};
Stack Class -
class Stack
{
Data Member -
Node *top // (pointer in C++)
Member Functions(methods) -
isEmpty() // check if stack empty
push(Node n) // push node on stack
pop() // pop top node from stack
peek() // get top node of stack
count() // get no of nodes in stack
checkIfNodeExist(Node n)
// check if node with same key exist
void display() // print all nodes in the stack
}
n1(5, 10, NULL)
n2(5, 25, )
Stack using Singly Linked List
TOP
Stack s
Computer Memory
#100 #102 #104 #106
#200 #202 #204 #206
#300 #302 #304 #306
16:37 / 50:09

## Implementing STACK using SINGLY LINKED LIST

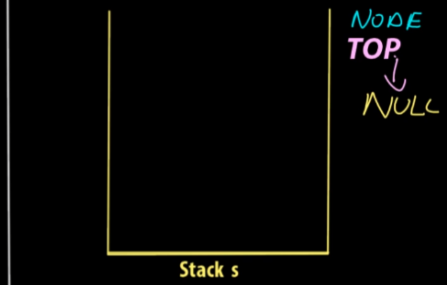
**Main Function -**

```
int main() {
Stack s
Node n1(1, 45),
  n2(3, 6), n3(7, 77),
  n4(2, 54), n5(9, 14)
s.isEmpty() -> TRUE
s.push(n1)
s.push(n2)
s.push(n3)
s.count()
s.pop()
s.pop()
s.peek()
s.count()
s.push(n4)
s.push(n5)
s.count()
s.display()
}
```

**isEmpty() -**

```
{
  1.if top-> NULL
       then true
  2.else
       false
}
```

**Stack using Singly Linked List**




**Computer Memory**

| #100 n1 | #102 | #104 n4 | #106 s |
|---|---|---|---|
| 1 45 NULL | | 2 54 NULL | *top NULL |
| #200 n2 | #202 n3 | #204 | #206 |
| 3 6 NULL | 7 77 NULL | | |
| #300 | #302 | #304 n5 | #306 |
| | | 9 14 NULL | |

## Implementing STACK using SINGLY LINKED LIST

**Main Function -**

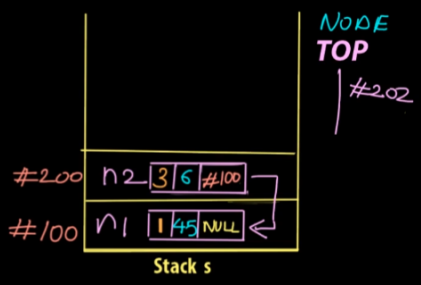
```
int main() {
Stack s
Node n1(1, 45),
  n2(3, 6), n3(7, 77),
  n4(2, 54), n5(9, 14)
s.isEmpty() -> TRUE
s.push(n1)
s.push(n2)
s.push(n3)
s.count()
s.pop()
s.pop()
s.peek()
s.count()
s.push(n4)
s.push(n5)
s.count()
s.display()
}
```

**push(Node n) -**

```
{
1.if top -> NULL
     then top = n
2.else if checkIfNodeExist(n)
     then print("Same key not
                allowed")
3.else
     Node temp = top;
     top = n;
     n->next=temp;
     print("Node  PUSHED")
}
```

temp= #200

**Stack using Singly Linked List**




**Computer Memory**

| #100 n1 | #102 | #104 n2 | #106 s |
|---|---|---|---|
| 1 45 NULL | | 2 54 NULL | *top #202 |
| #200 n2 | #202 n3 | #204 | #206 |
| 3 6 #100 | 7 77 #200 | | |
| #300 | #302 | #304 n5 | #306 |
| | | 9 14 NULL | |

29:42 / 50:09

## Implementing STACK using SINGLY LINKED LIST

**Main Function -**

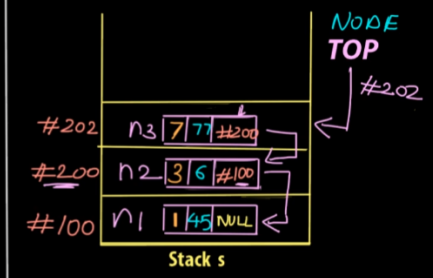
```
int main() {
Stack s
Node n1(1, 45),
  n2(3, 6), n3(7, 77),
  n4(2, 54), n5(9, 14)
s.isEmpty() -> TRUE
s.push(n1)
s.push(n2)
s.push(n3)
s.count() // 3
s.pop()
  s.pop()
  s.peek()
  s.count()
  s.push(n4)
  s.push(n5)
  s.count()
  s.display()
}
```

**count() -**

```
{
 1.count=0
2.Node*temp=top
3.while->temp!=NULL
  {
3.1count++
3.2temp=temp->next
  }
4.return count
}
```

temp = NULL

**Stack using Singly Linked List**




Computer Memory


## Implementing STACK using SINGLY LINKED LIST

**Main Function -**

```
int main() {
Stack s
Node n1(1, 45),
  n2(3, 6), n3(7, 77),
  n4(2, 54), n5(9, 14)
s.isEmpty() -> TRUE
s.push(n1)
s.push(n2)
s.push(n3)
s.count() // 3
s.pop()
s.pop()
s.peek()
  s.count()// 1
s.push(n4)
s.push(n5)
s.count() //3
  s.display()
}
```

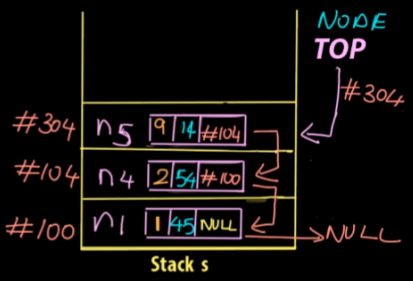
**pop() -**

```
{
  1.Node temp = NULL
  2.if -> s.isEmpty() then
     2.1print "stack underflow"
     2.2return temp
  3.else
     3.1temp=top
     3.2top = top->next
  4.return temp
 }
```

**peek() -**

```
{
  1.if -> s.isEmpty() then
     1.1print "stack underflow"
     1.2return NULL
  2.else
     2.1return top
}
```

**Stack using Singly Linked List**




Computer Memory

## Implementing STACK using SINGLY LINKED LIST

**Main Function -**

```
int main() {
Stack s
Node n1(1, 45),
  n2(3, 6), n3(7, 77),
  n4(2, 54), n5(9, 14)
s.isEmpty() -> TRUE
s.push(n1)
s.push(n2)
s.push(n3)
s.count() // 3
s.pop()
s.pop()
s.peek()
  s.count()// 1
s.push(n4)
s.push(n5)
s.count() //3
s.display()
}
```

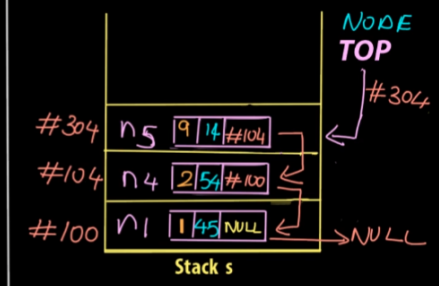
**display() -**

```
{
1.temp=top
2.LOOP: while->temp!=NULL
    2.1 print(temp->key","
              temp->data")
    2.2 temp=temp->next
  END LOOP
}
```

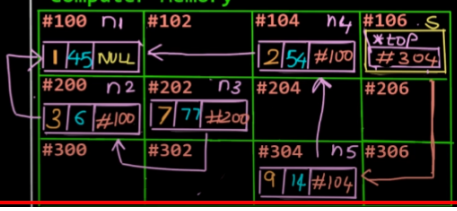
temp = NULL

n5 : (9,14), n4 : (2,54), (1,45)

Stack using Singly Linked List




Computer Memory




## Implementing STACK using SINGLY LINKED LIST

**Main Function -**

```
int main() {
Stack s
Node n1(1, 45),
  n2(3, 6), n3(7, 77),
  n4(2, 54), n5(9, 14)
s.isEmpty() -> TRUE
s.push(n1)
s.push(n2)
s.push(n3)
s.count() // 3
s.pop()
s.pop()
s.peek()
  s.count()// 1
s.push(n4)
s.push(n5)
s.count() //3
s.display()
}
```

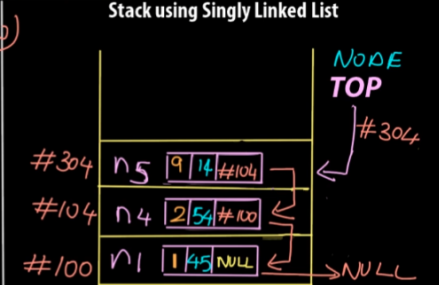
**checkIfNodeExist(Node n) -** n6 (2,500)

```
{
1.Node temp=top
2.exist=false
3.LOOP: while->temp!=NULL
  3.1 if temp->key == n->key
        then   2 == 2
    3.1.1 exist=true
    3.1.2 break
  3.2 temp=temp->next
  END LOOP
4.return exist
}
```

temp = #104

exist = true

Stack using Singly Linked List




Computer Memory