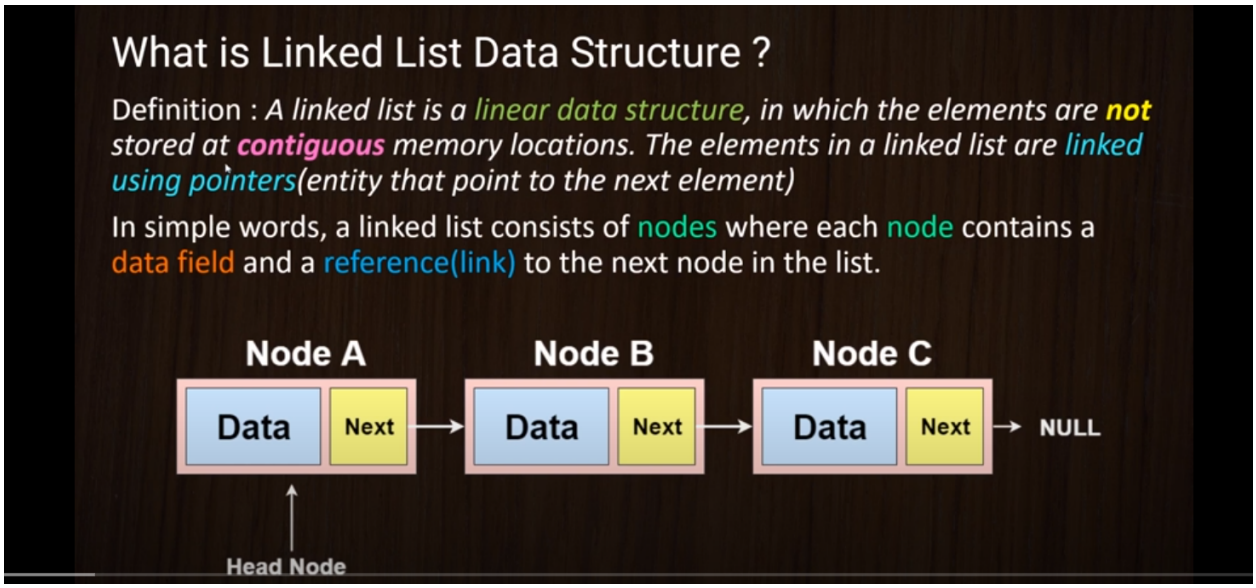
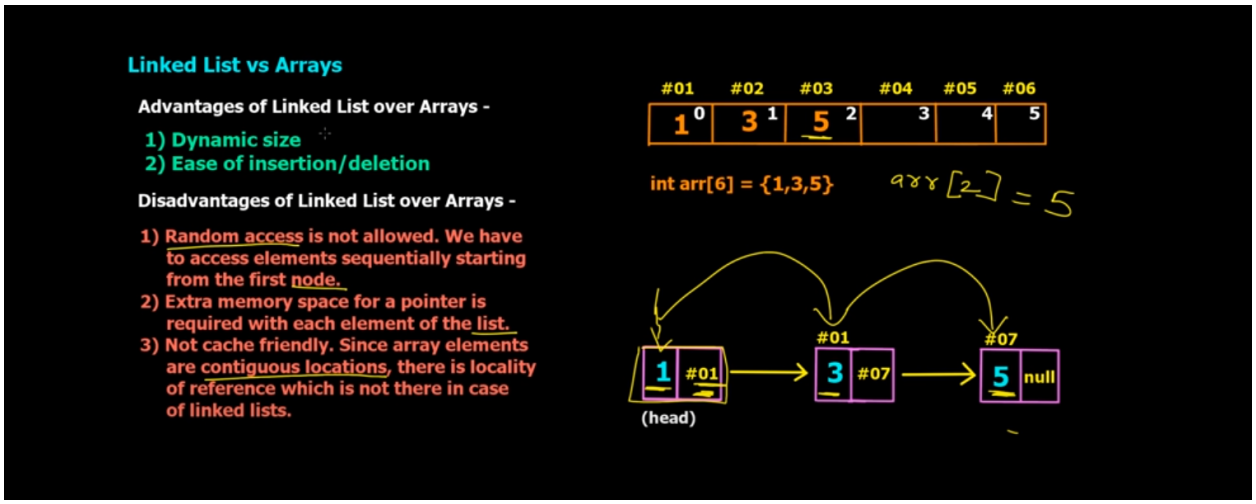
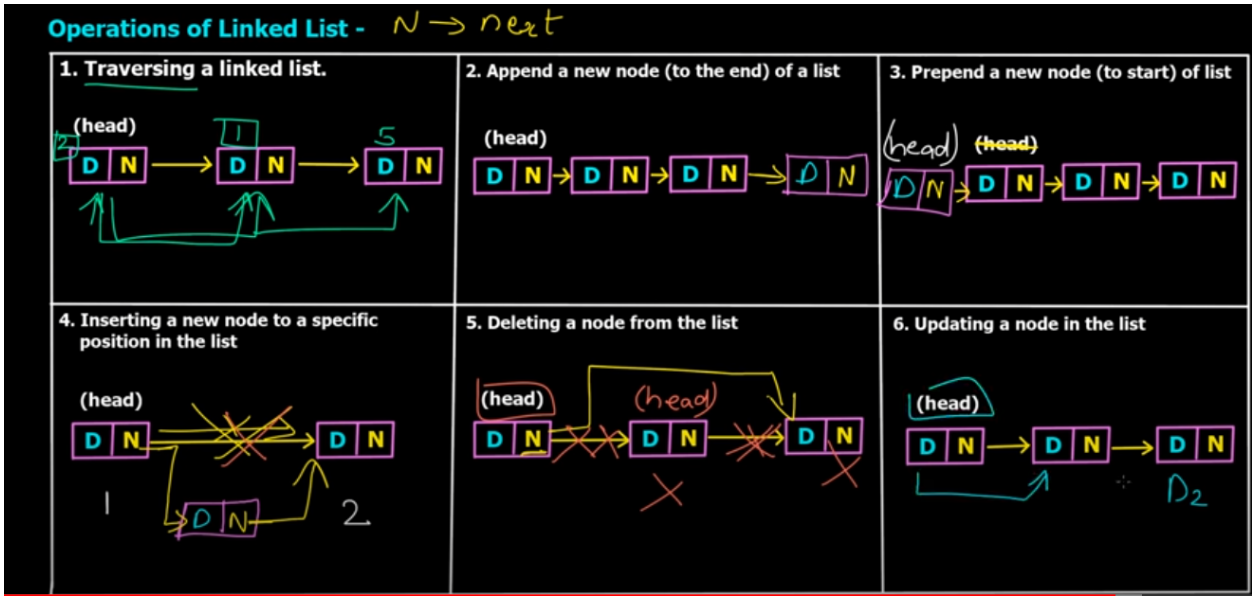
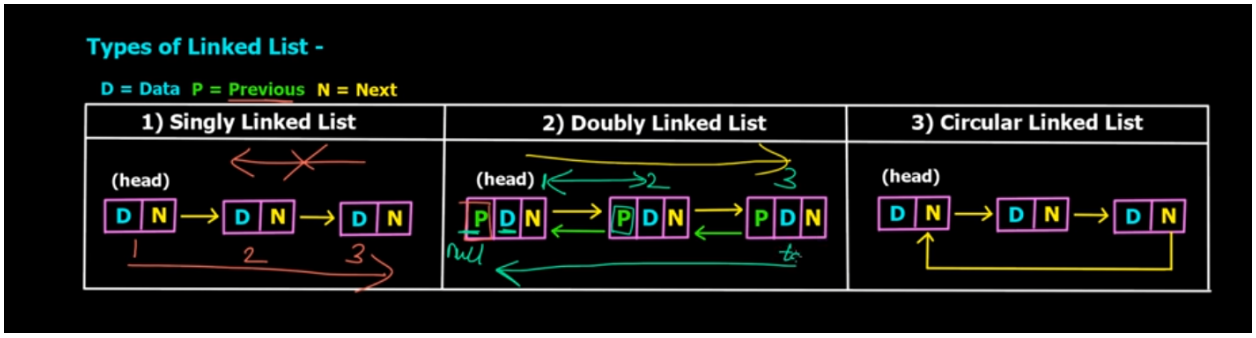
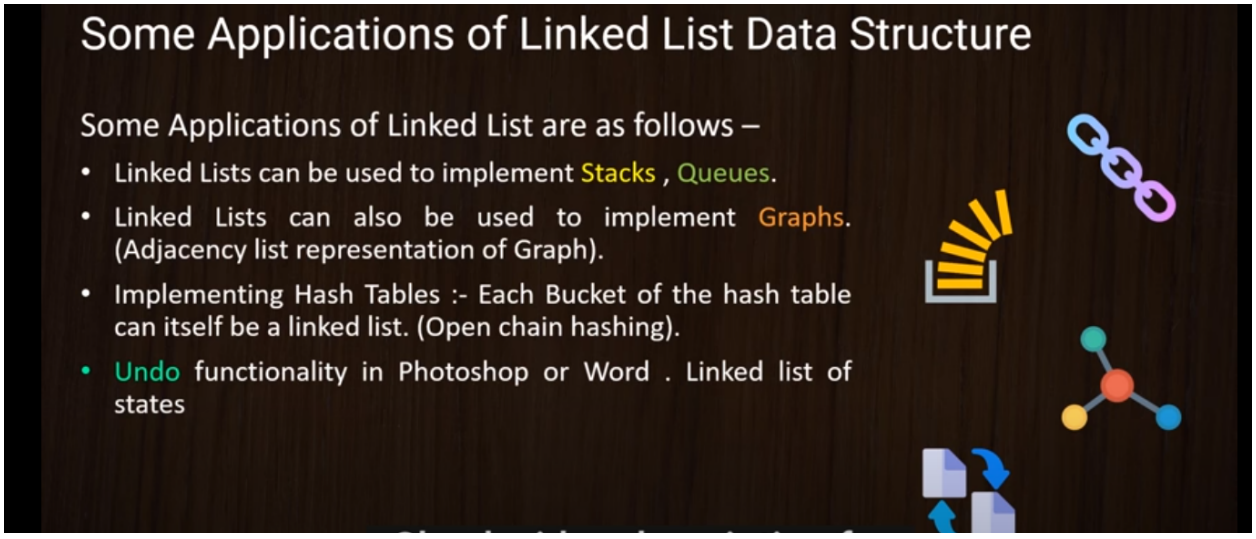
What is a linked list?



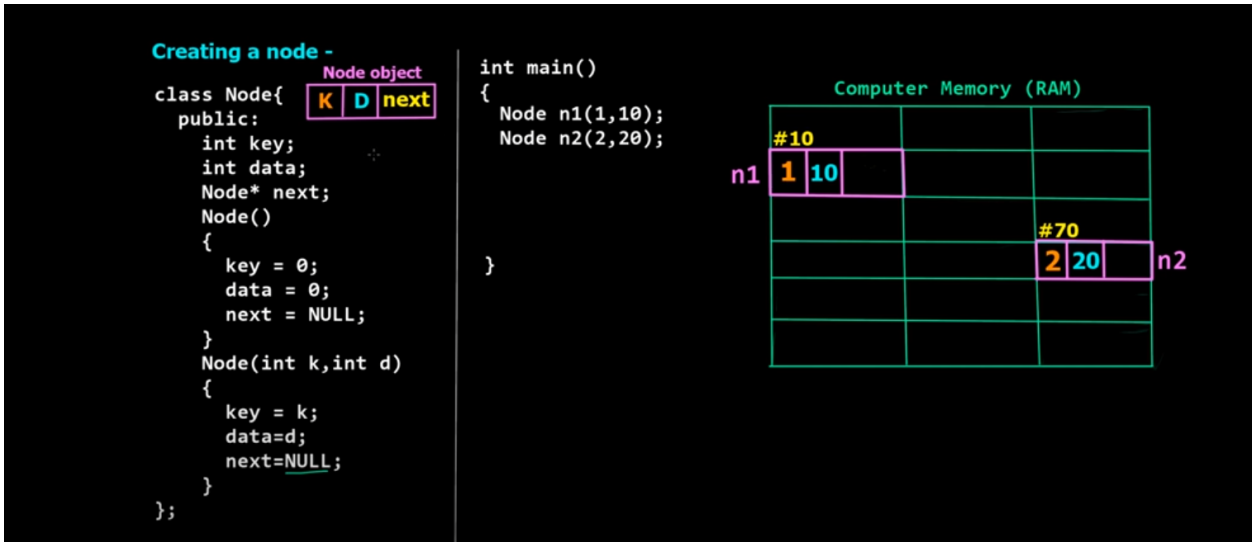


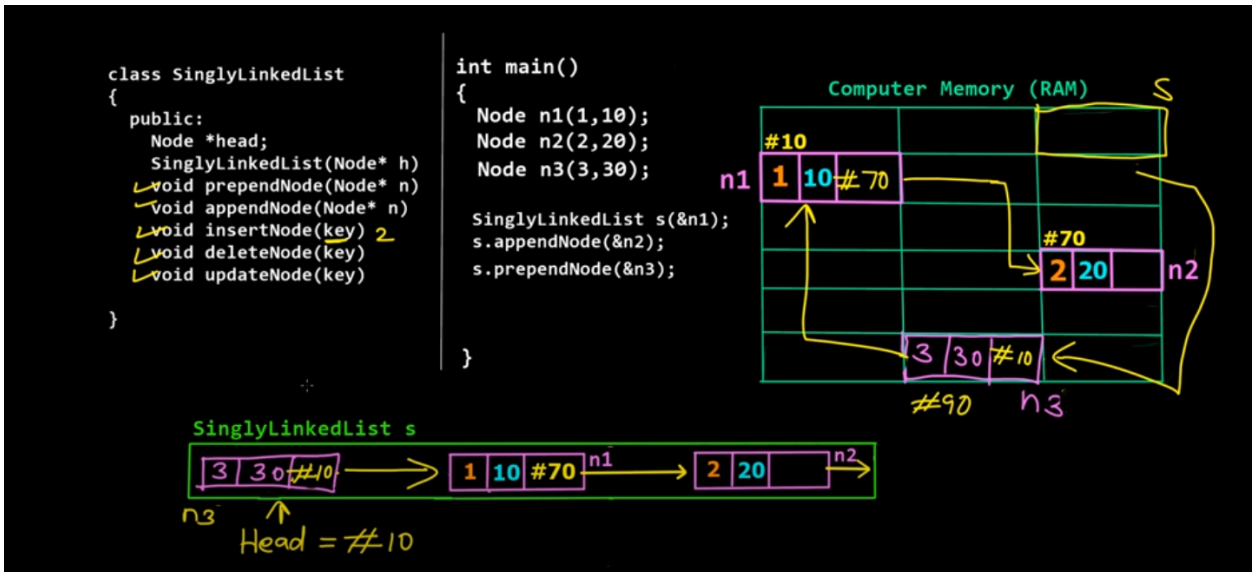

Operations of Linked List - N → next
1. Traversing a linked list.
(head)
2. Append a new node (to the end) of a list
(head)
3. Prepend a new node (to start) of list
(head)
(head)
4. Inserting a new node to a specific position in the list
(head)
5. Deleting a node from the list
(head)
(head)
6. Updating a node in the list
(head)


Types of Linked List -
D = Data P = Previous N = Next
1) Singly Linked List
2) Doubly Linked List
3) Circular Linked List
(head)
(head)
(head)

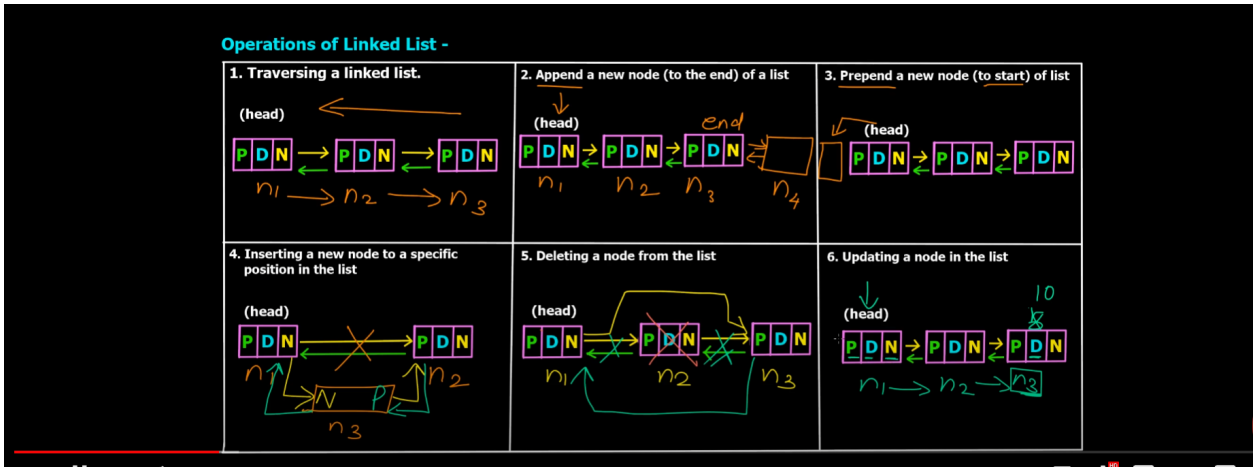

Some Applications of Linked List Data Structure
Some Applications of Linked List are as follows –
• Linked Lists can be used to implement Stacks , Queues.
• Linked Lists can also be used to implement Graphs. (Adjacency list representation of Graph).
• Implementing Hash Tables :- Each Bucket of the hash table can itself be a linked list. (Open chain hashing).
• Undo functionality in Photoshop or Word . Linked list of states

Node





Doubly Linked List



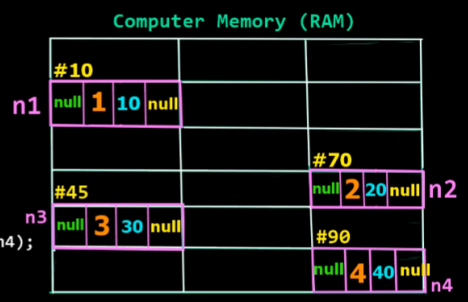
Creating a node -

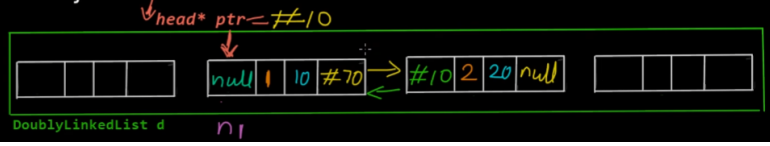
Node Obj

Append node algorithm/pseudo code

1. Get the new node.
2. Check if node exists with same key
2a. if true, abort
2b. if false continue
3a. if head == NULL then append at start
3b. else traverse to the end of the list
4. Append new node at the end

```
int main()
{
  Node n1(1,10);
  Node n2(2,20);
  Node n3(3,30);
  Node n4(4,40);
  DoublyLinkedList d
  d.appendNode(&n1);
  d.appendNode(&n2);
  d.prependNode(&n3);
  d.insertNodeAfter(1,&n4);
  d.deleteNode(3);
  d.updateNode(4,55);
}
```





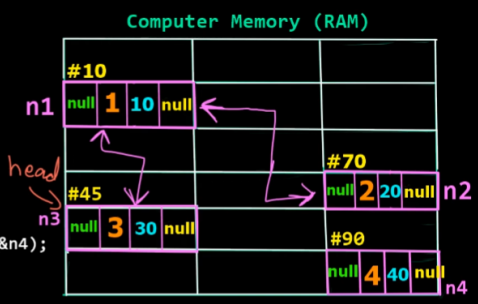
Creating a node -

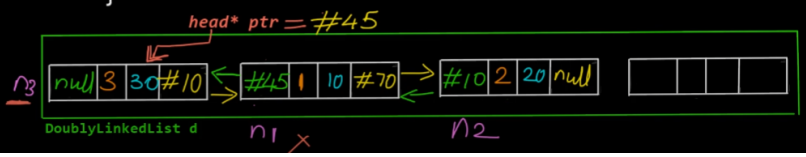
Node Obj

Prepend node algorithm/pseudo code

1. Get the new Node.
2. Check if node exists with same key
2a. if true, abort
2b. if false continue
3. head->previous = new_node (n3)
4. new_node->next = head
5. head = n (&n3) = new_node

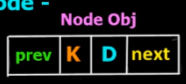
```
int main()
{
  Node n1(1,10);
  Node n2(2,20);
  Node n3(3,30);
  Node n4(4,40);
  DoublyLinkedList d
  d.appendNode(&n1);
  d.appendNode(&n2);
  d.prependNode(&n3);
  d.insertNodeAfter(1,&n4);
  d.deleteNode(3);
  d.updateNode(4,55);
}
```





Creating a node -

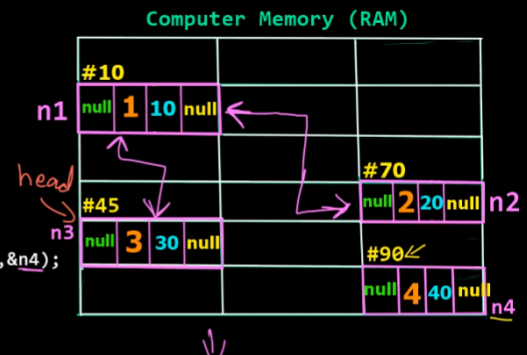
Node Obj

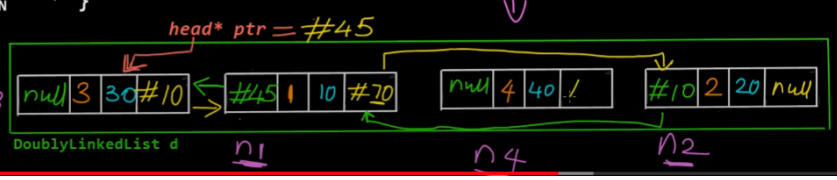


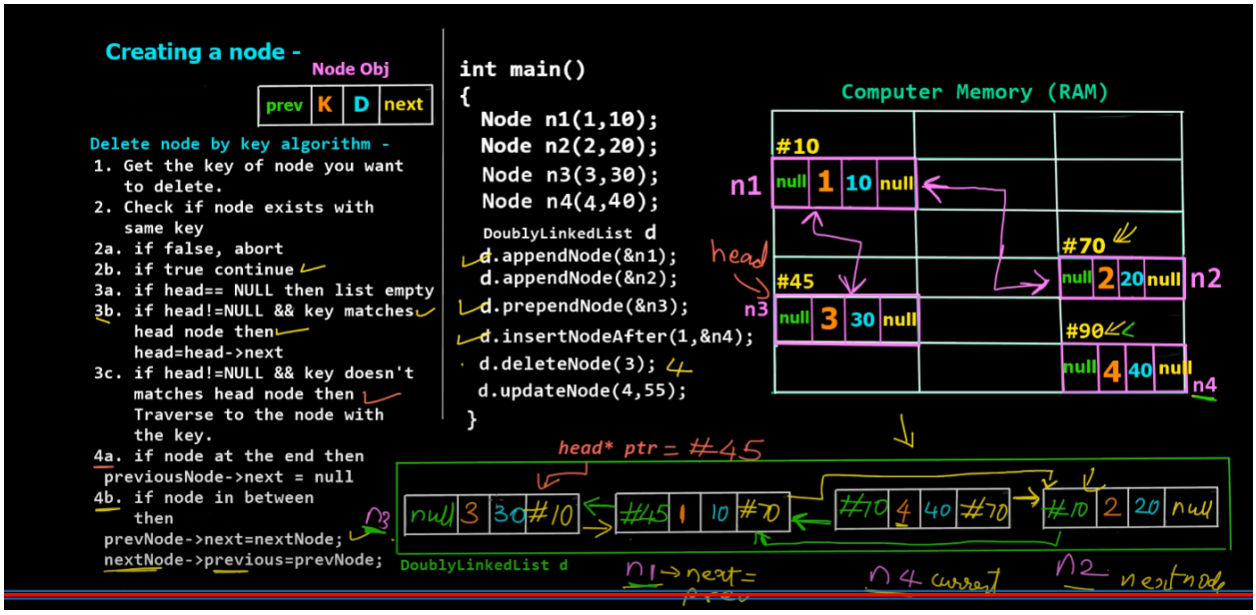
Insert node node after algorithm/pseudo code

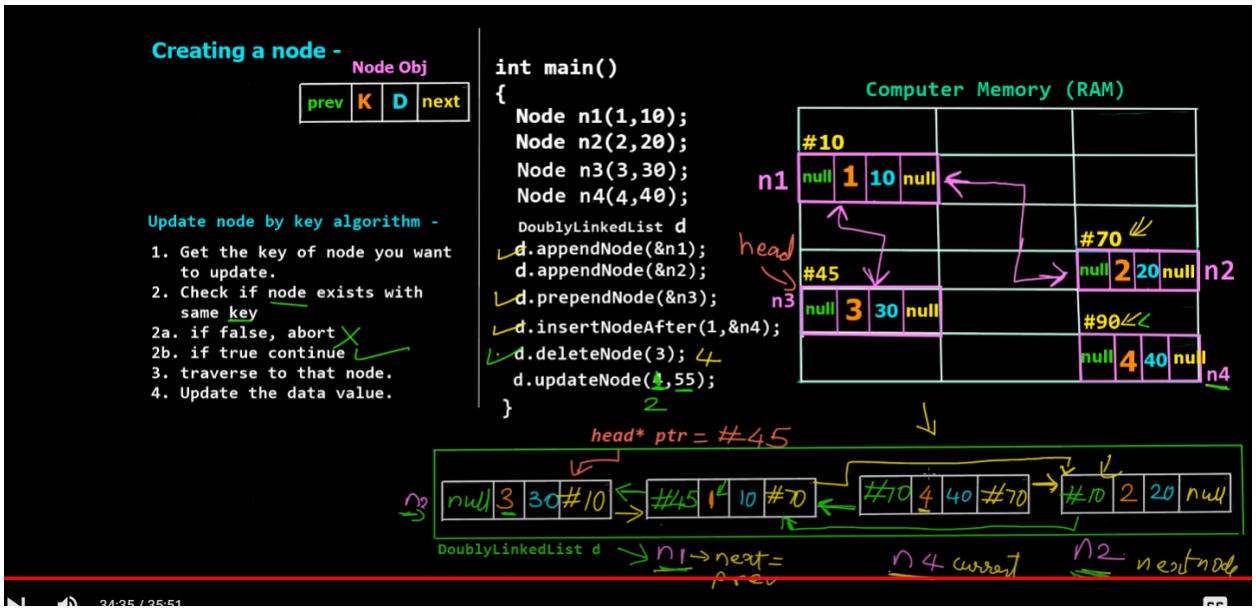
1. Get the new Node & the key of the node in the list after which you want to link this new node.
2. Check if node exists with same key as new node
2a. if true, abort
2b. if false continue
3. Check if node exists with the key entered by user
3a. if false, abort (N=n1)
3b. if true continue (new_N= n4)
4. access node N after which you want to append new node new_N
4a. if N is at the end then
N->next = new_N
new_N->previous = N;
4b. if N is in between.
new_N->next = N->next
N->next->previous = new_N
new_N->previous = N
n->next = new_N

```
int main()
{
  Node n1(1,10);
  Node n2(2,20);
  Node n3(3,30);
  Node n4(4,40);
  DoublyLinkedList d
  d.appendNode(&n1);
  d.appendNode(&n2);
  d.prependNode(&n3);
  d.insertNodeAfter(1,&n4);
  d.deleteNode(3);
  d.updateNode(4,55);
}
```

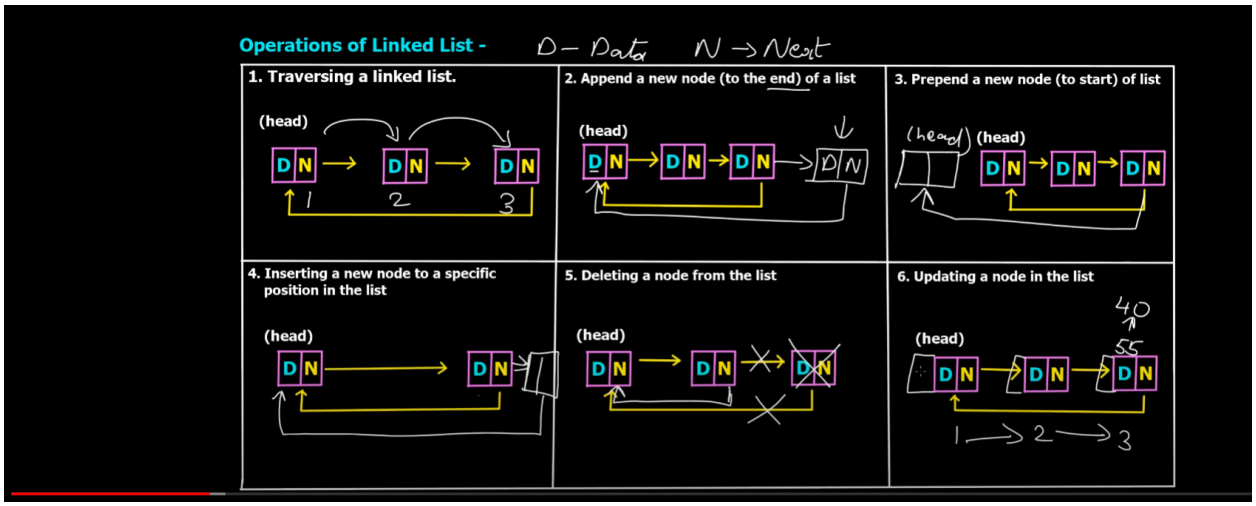


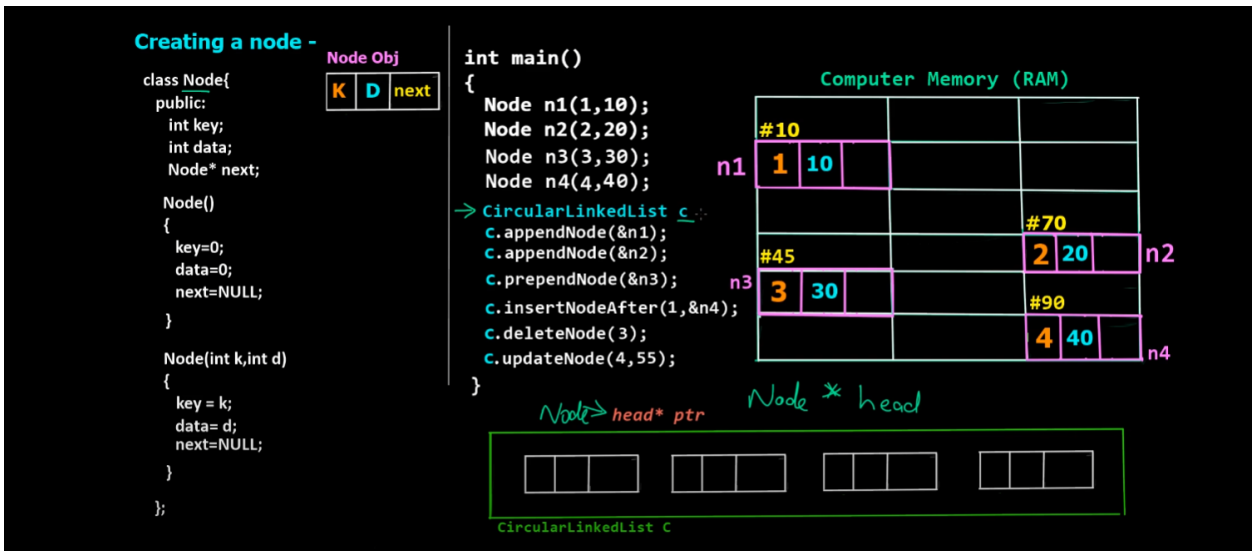


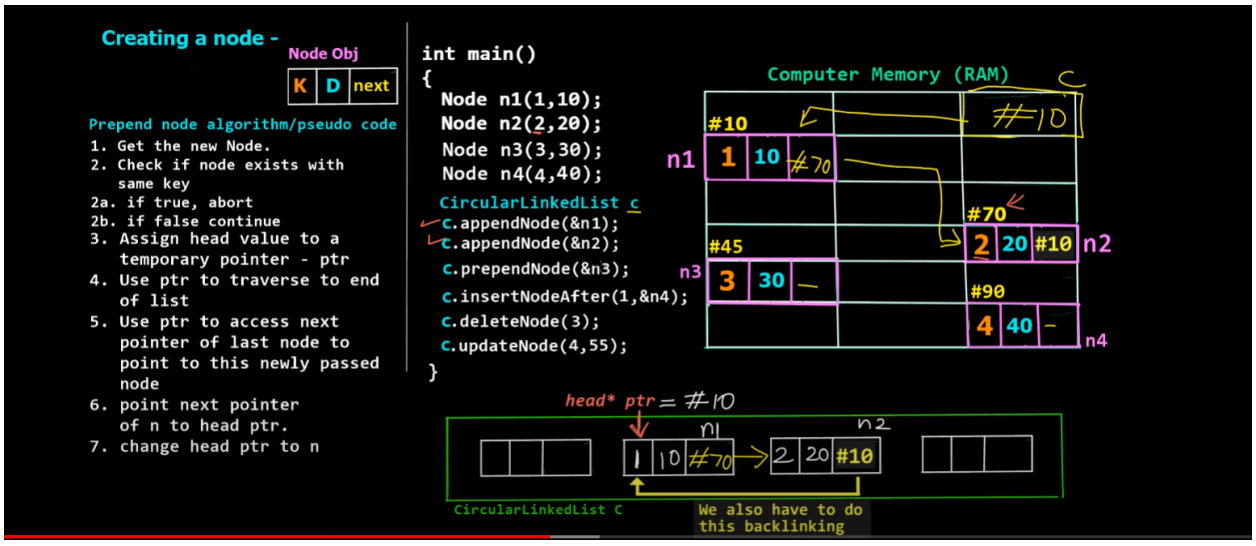

Creating a node -
Node Obj
prev K D next
Delete node by key algorithm -
1. Get the key of node you want to delete.
2. Check if node exists with same key
2a. if false, abort
2b. if true continue
3a. if head== NULL then list empty
3b. if head!=NULL && key matches head node then head=head->next
3c. if head!=NULL && key doesn't matches head node then Traverse to the node with the key.
4a. if node at the end then previousNode->next = null
4b. if node in between then
prevNode->next=nextNode;
nextNode->previous=prevNode;
int main()
{
Node n1(1,10);
Node n2(2,20);
Node n3(3,30);
Node n4(4,40);
DoublyLinkedList d
d.appendNode(&n1);
d.appendNode(&n2);
d.prependNode(&n3);
d.insertNodeAfter(1,&n4);
d.deleteNode(3);
d.updateNode(4,55);
}
Computer Memory (RAM)
#10
n1
null 1 10 null
head
#45
n3
null 3 30 null
#70
null 2 20 null
n2
#90
null 4 40 null
n4
head* ptr = #45
DoublyLinkedList d
n1->next=
n4 current
n2 nextNode


Creating a node -
Node Obj
prev K D next
Update node by key algorithm -
1. Get the key of node you want to update.
2. Check if node exists with same key
2a. if false, abort
2b. if true continue
3. traverse to that node.
4. Update the data value.
int main()
{
Node n1(1,10);
Node n2(2,20);
Node n3(3,30);
Node n4(4,40);
DoublyLinkedList d
d.appendNode(&n1);
d.appendNode(&n2);
d.prependNode(&n3);
d.insertNodeAfter(1,&n4);
d.deleteNode(3);
d.updateNode(4,55);
}
Computer Memory (RAM)
#10
n1
null 1 10 null
head
#45
n3
null 3 30 null
#70
null 2 20 null
n2
#90
null 4 40 null
n4
head* ptr = #45
DoublyLinkedList d
n1->next=
n4 current
n2 nextNode

Circular Linked List







Creating a node -

Node Obj

K	D	next

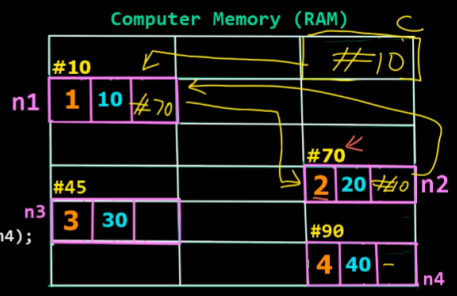
Insert node node after algorithm -

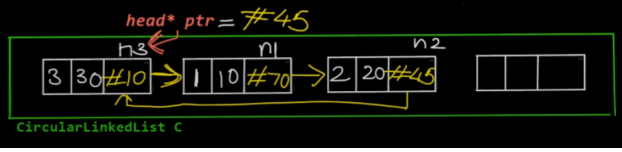
1. Get the new Node & the key of the node in the list after which you want to link this new node.
2. Check if node exists with same key as new node
2a. if true, abort
2b. if false continue
3. Check if node exists with the key entered by user
3a. if false, abort
3b. if true continue
4. access node N after which you want to append new node new_N

4a. If node to be inserted at the end then
new_node->next = head
previous_node->next = new_node

4b. if node to be inserted in between then
new_node->next = previous_node->next
previous_node->next = new_node

```
int main()
{
  Node n1(1,10);
  Node n2(2,20);
  Node n3(3,30);
  Node n4(4,40);
  CircularLinkedList c
  c.appendNode(&n1);
  c.appendNode(&n2);
  c.prependNode(&n3);
  c.insertNodeAfter(1,&n4);
  c.deleteNode(3);
  c.updateNode(4,55);
}
```





Creating a node -

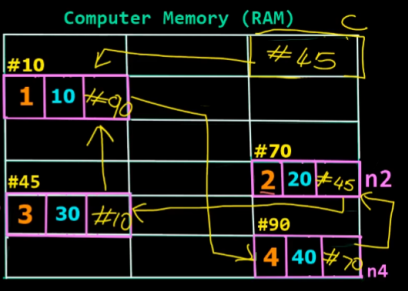
Node Obj

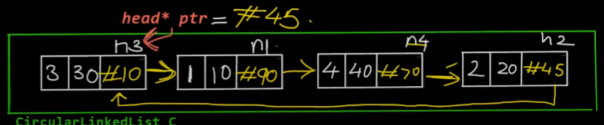
K	D	next

Delete node by key algorithm -

1. Get the key of node you want to delete.
2. Check if node exists with same key
2a. if false, abort
2b. if true continue
3a. if head== NULL then list empty
3b. if head!=NULL && key matches head node then
head=head->next
last_node->next = head
3c. if head!=NULL && key doesn't matches head node then
Traverse to the node with the key.(ptr)
4a. if node at the end then
previousNode->next = head
4b. if node in between then
prevNode->next=ptr->next

```
int main()
{
  Node n1(1,10);
  Node n2(2,20);
  Node n3(3,30);
  Node n4(4,40);
  CircularLinkedList c
  c.appendNode(&n1);
  c.appendNode(&n2);
  c.prependNode(&n3);
  c.insertNodeAfter(1,&n4);
  c.deleteNode(3);
  c.updateNode(4,55);
}
```





Creating a node -

Node Obj

K	D	next

Update node by key algorithm -

1. Get the key of node you want to update.
2. Check if node exists with same key
2a. if false, abort
2b. if true continue
3. traverse to that node.
4. Update the data value.

```
int main()
{
  Node n1(1,10);
  Node n2(2,20);
  Node n3(3,30);
  Node n4(4,40);
  CircularLinkedList c
  c.appendNode(&n1);
  c.appendNode(&n2);
  c.prependNode(&n3);
  c.insertNodeAfter(1,&n4);
  c.deleteNode(3);
  c.updateNode(4,55);
}
```