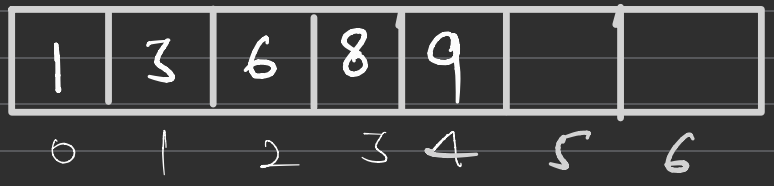
Doing the best at this moment, puts you in the best place for the next moment.

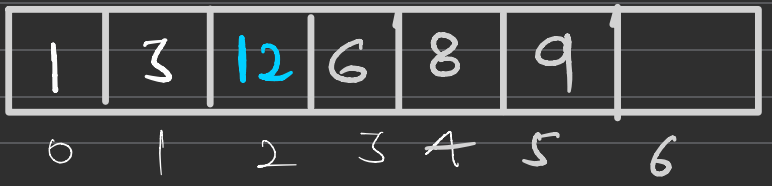
If we have Array, then Why Linked List

dis-advantages

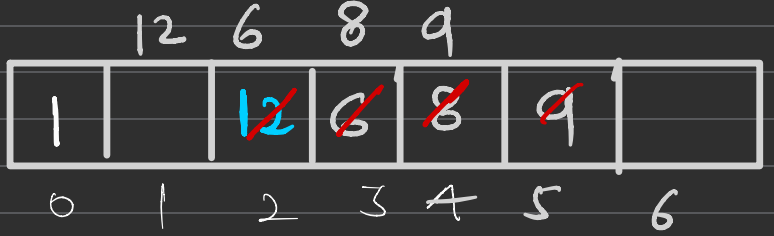


Element = 12

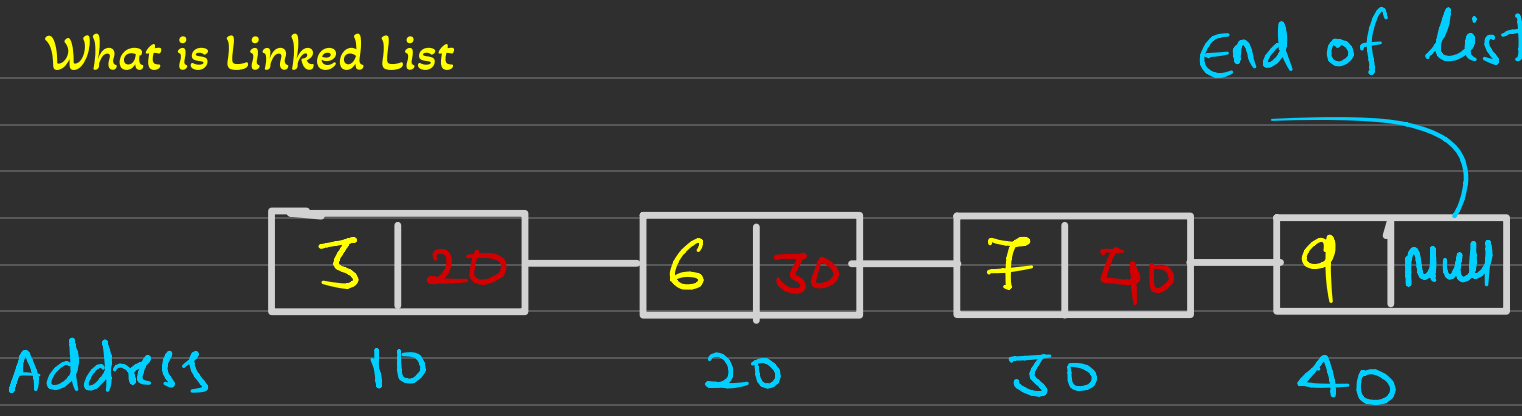
(insert at index 2)



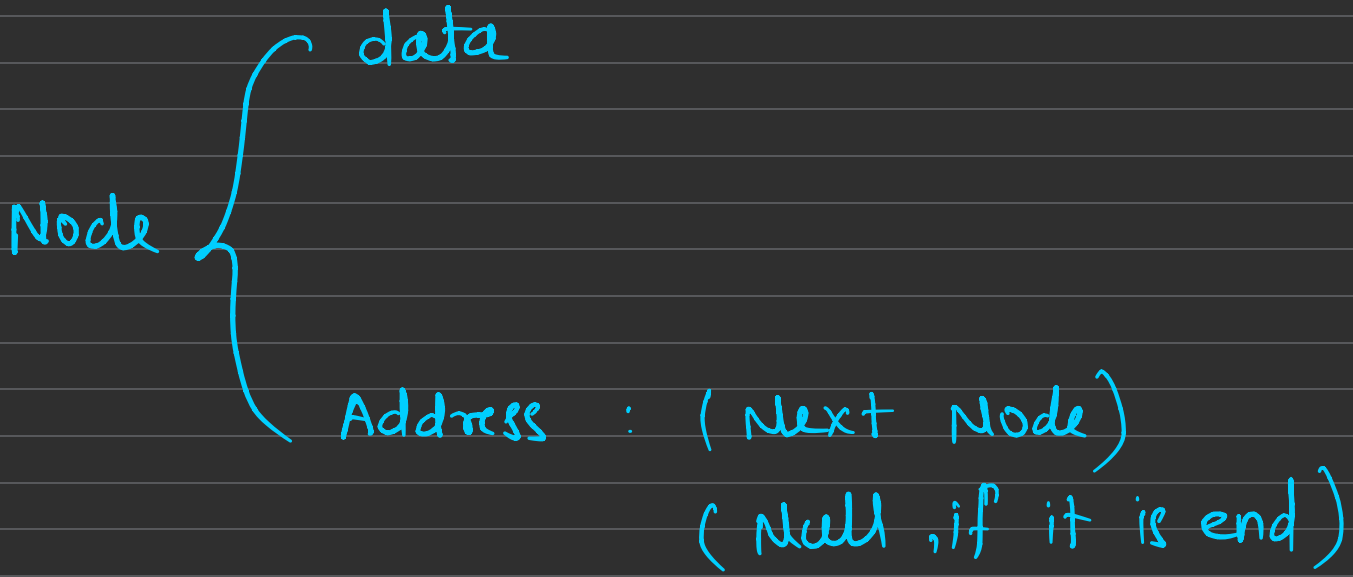
Please delete element from index 1



What is Linked List



Collection of data, which is represented in term of Node.



Head / Start Node

- if head == null → list is Empty
- Not to change head pointer

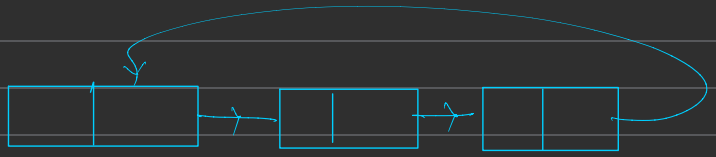
- Head indicates start of list

Types of Linked List

Singly Linked List

Doubly Linked List

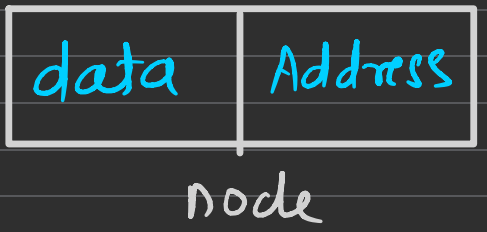
Circular Linked List



- Types of operation:

* Insertion
  - start
  - end
  - Position
* Deletion
  - start
  - end
  - Position
* Traversal

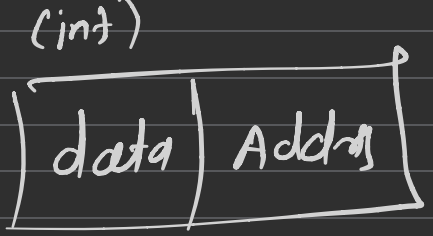
Structure of Linked List (SLL)



C++

```
class Node
{
    int data
    Node * next

};
```



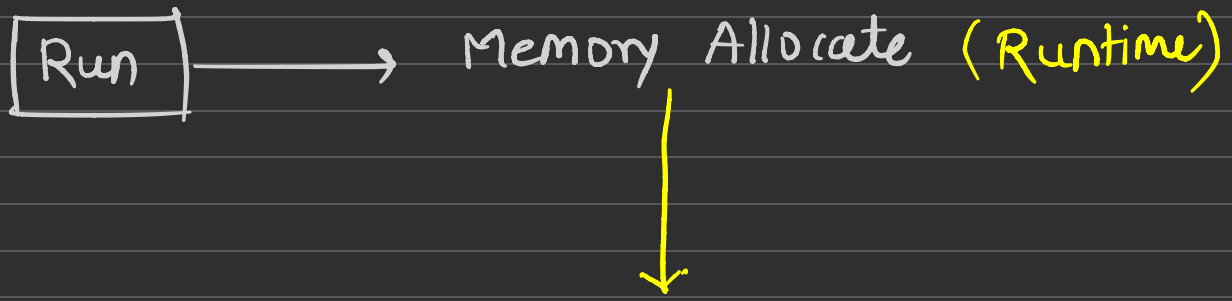
Java

```
class Node
{
    int data
    Node next

}
```

How to create Node

int x



* new keyword allocates memory Dynamic memory, from heap

```
Node *node = new Node()
```

Implementing SLL

```
class Node
{
    int data
    Node * next

    Node (int d)
    {
        data = d
        next = null
    }
}
```

```
Node *node = new Node(10)
```