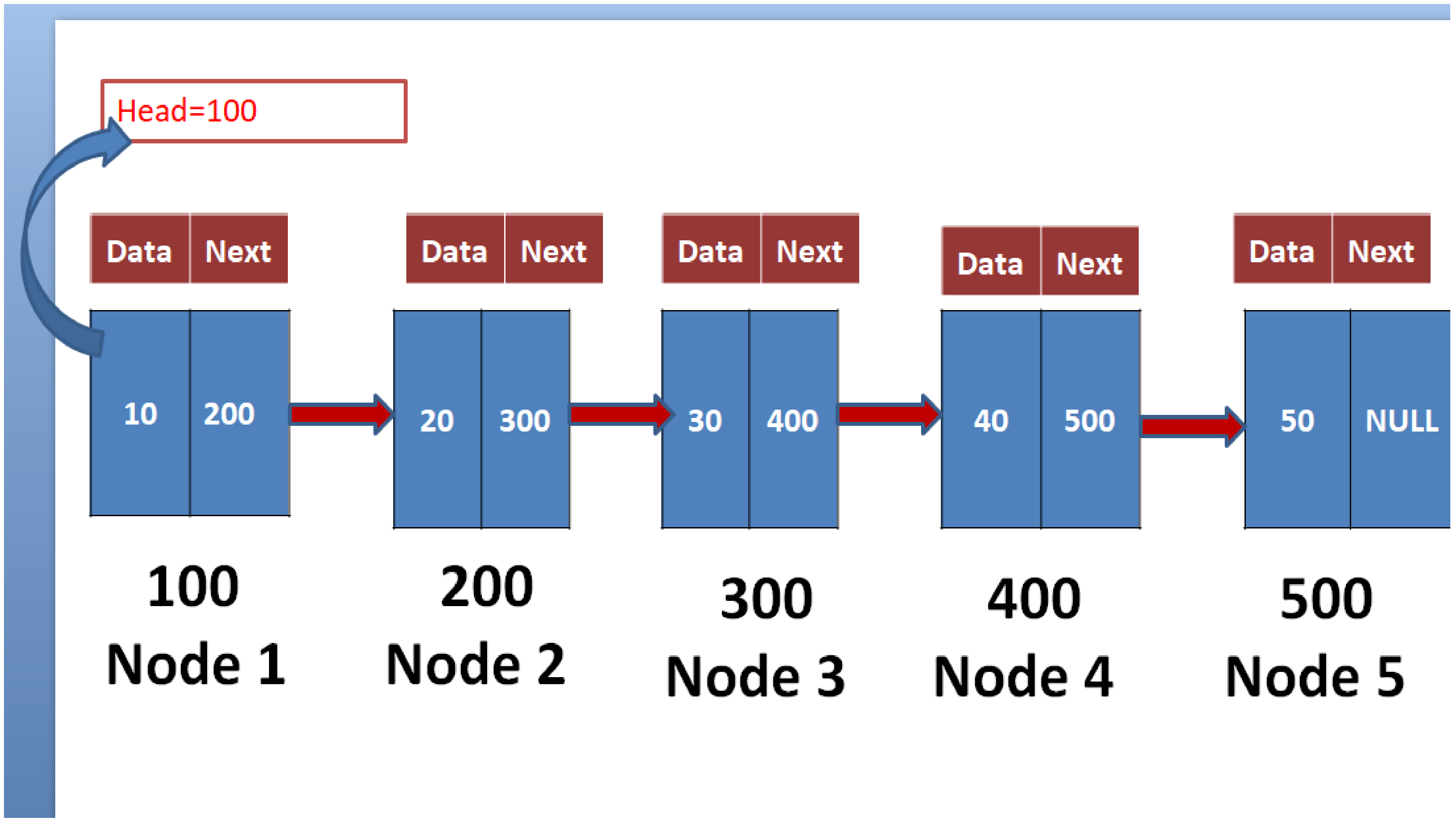
Single Linked List

Terminology: Head, Node, Data, Next, NULL
Operations: create(), insertion( ), deletion( ), Display(), Search( )

SLL


Head=100
Data
Next
10
200
Data
Next
20
300
Data
Next
30
400
Data
Next
40
500
Data
Next
50
NULL
100
Node 1
200
Node 2
300
Node 3
400
Node 4
500
Node 5

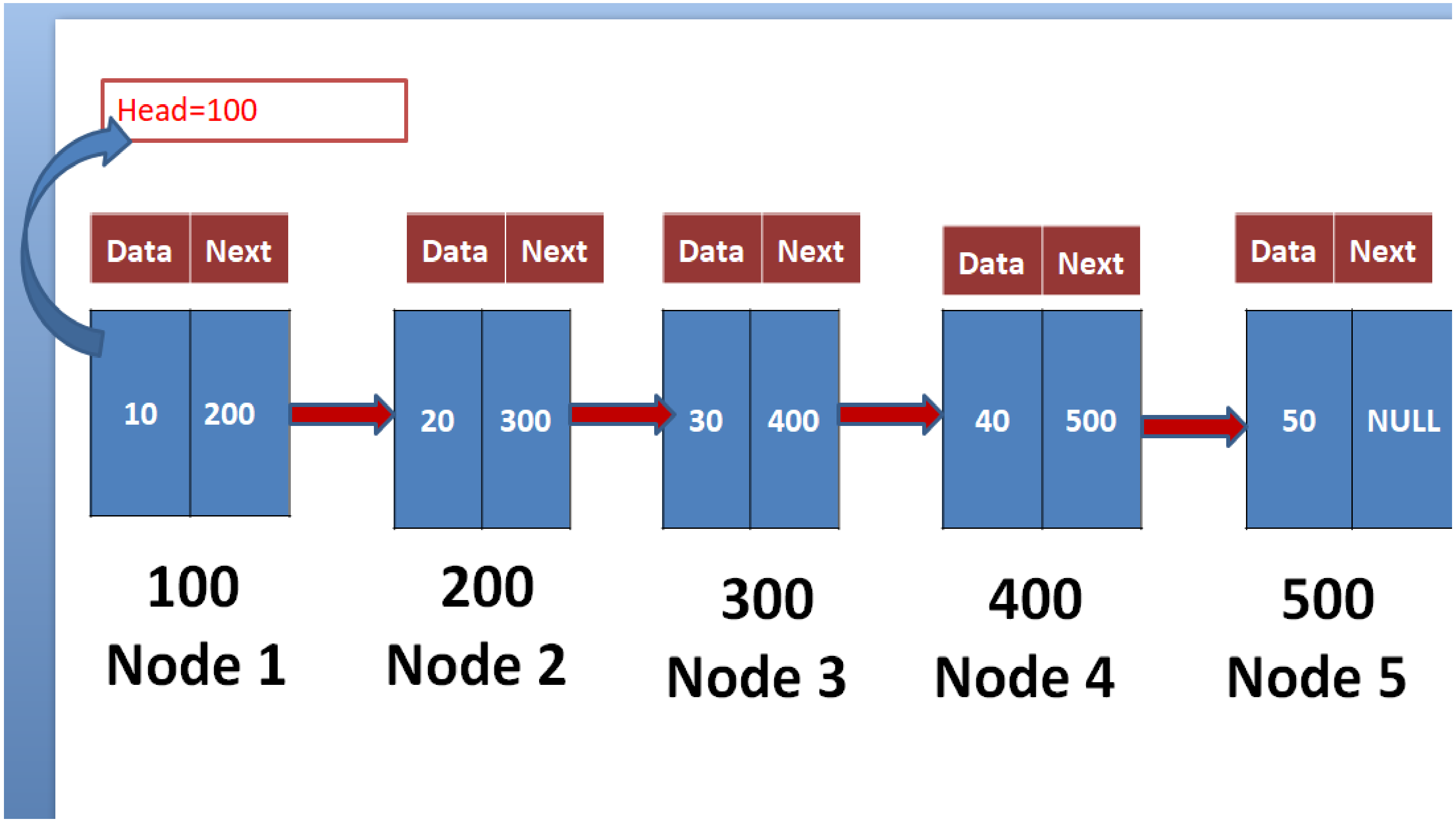
Content

- What is Single linked list
- What is node?
- Why are we using Single linked list instead of stack and Queue.
- Node Structure.
- Connecting nodes by address
- Operations on SLL.

What is Single linked list

- **List:** collection of number of elements
- **SLL:** SLL is linear Data Structure.
- It is also a collection of elements(nodes) but every element is linked with next element(node) by address.

Example picture



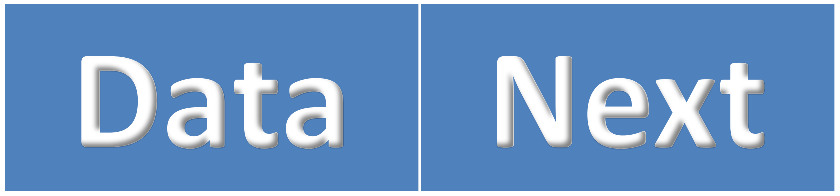
What is node?

- Every single element in a List is called “Node”.
- Node contains two fields

1) Data filed-it holds data(element value)

2) Next field- it holds address of next node

- Every node has it’s own address value in the memory



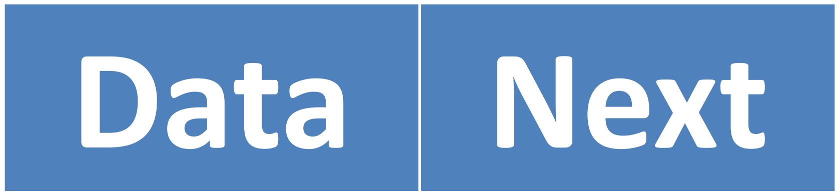
Node
100

Why are we using Single linked list instead of stack and Queue.

- Stack and Queue are linear DS, those are having limited elements (static size).
- But Linked Lists are having unlimited elements (Dynamic).
- Insertion at middle is not possible in Queue.
- But it is possible in stack, it takes more operations to perform

Node Structure

```
Structure Node
{
Int data;
Structure Node *Next;
}*head=NULL;
```

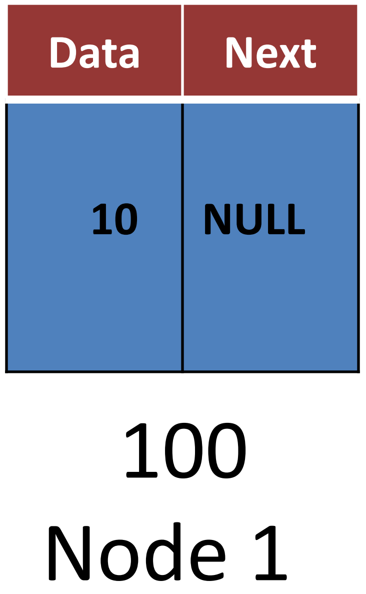


Node

100

Connecting nodes by address

1)Before creating first node :: Assign **Head=NULL**



```
Node1=(*struct Node)malloc(sizeof(*struct Node);

Node1->data=10;
Node1->next=NULL;
```

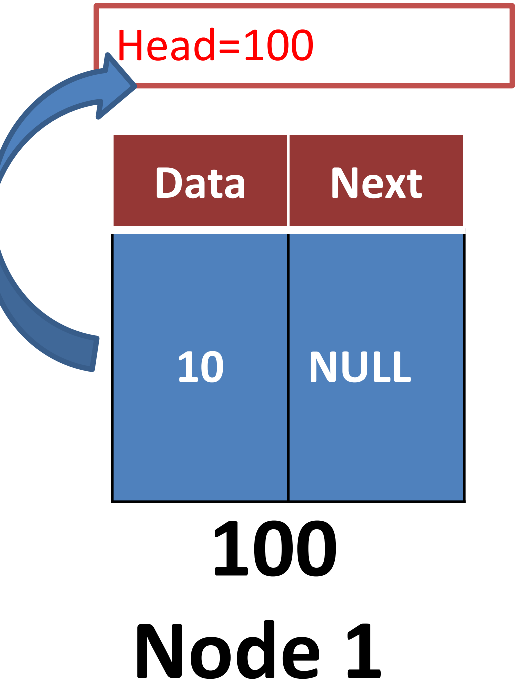
```
If(head==NULL)
{
Head=node1;
}
```

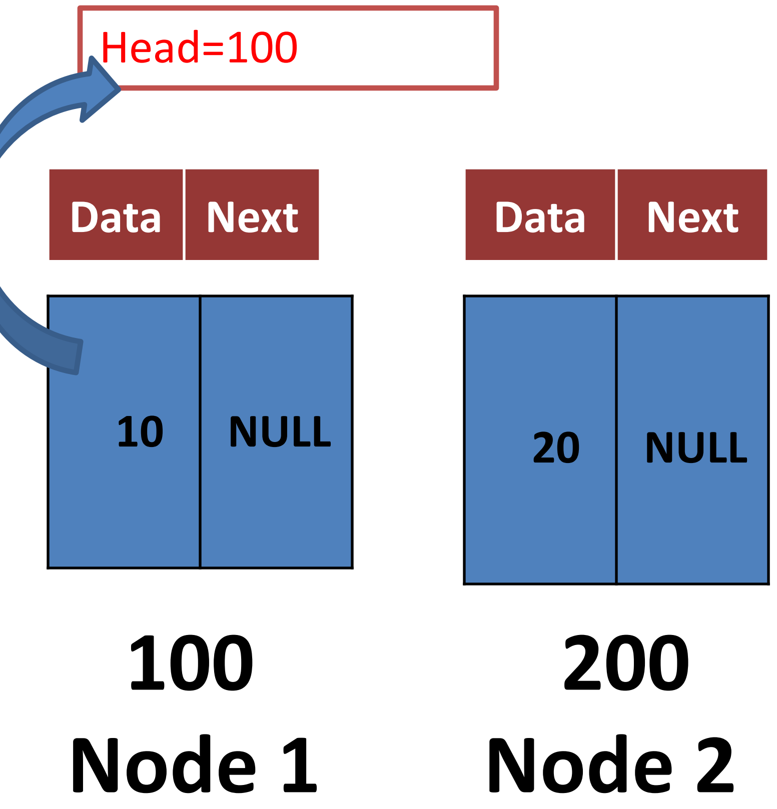
i.e:

```
100->data=10;
100->next=NULL;
```

(first node of list is called “Head” in SLL.

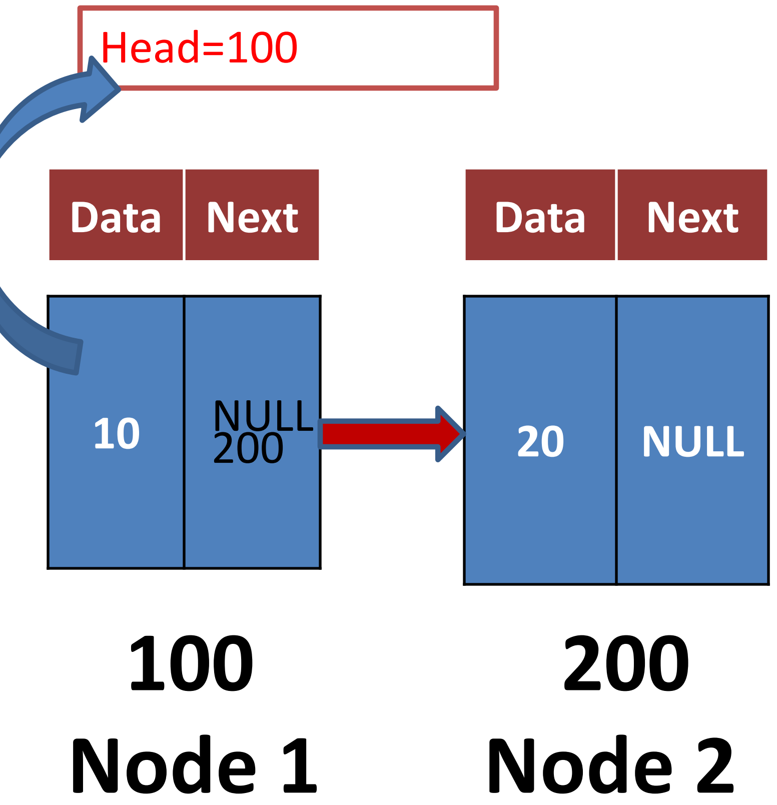
After creating first node Head=first node address
i.e **Head=100**



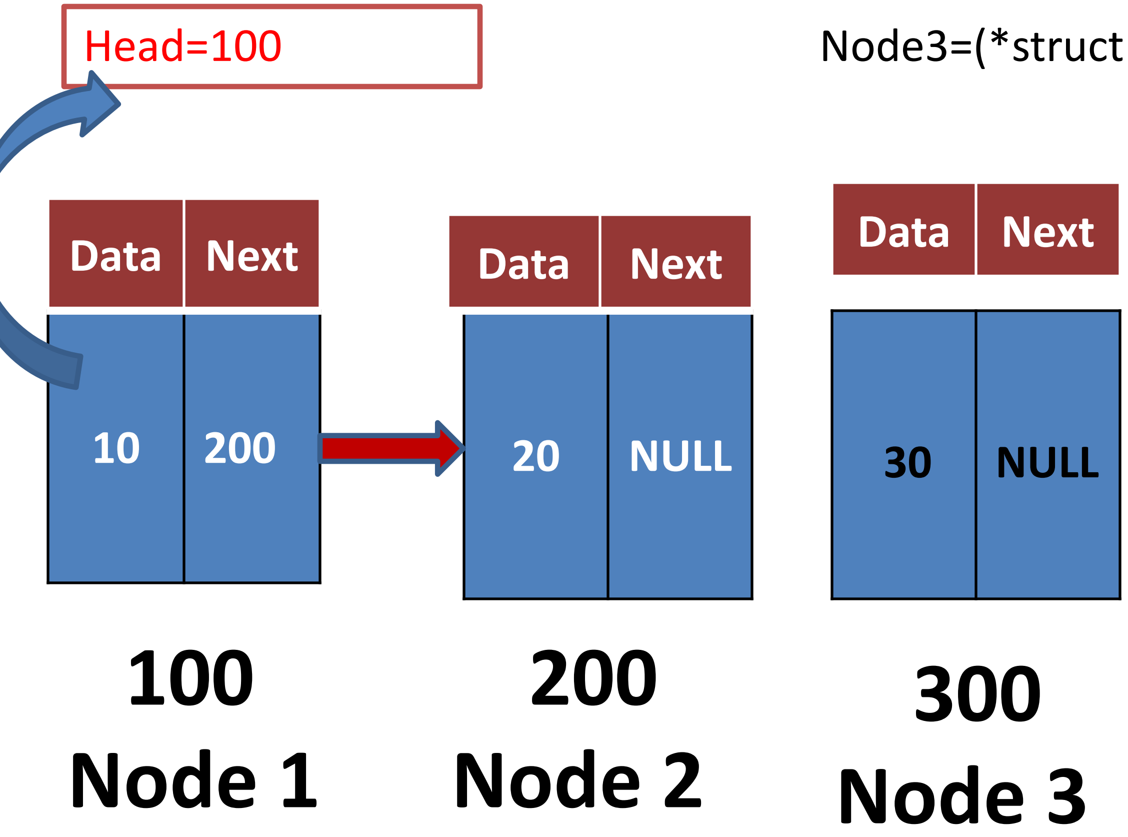


```
Node2=(*struct Node)malloc(sizeof(*struct Node);


Node2->data=20;
Node2->next=NULL;
```

```
If(head==NULL)
{
Head=node1;
}
```



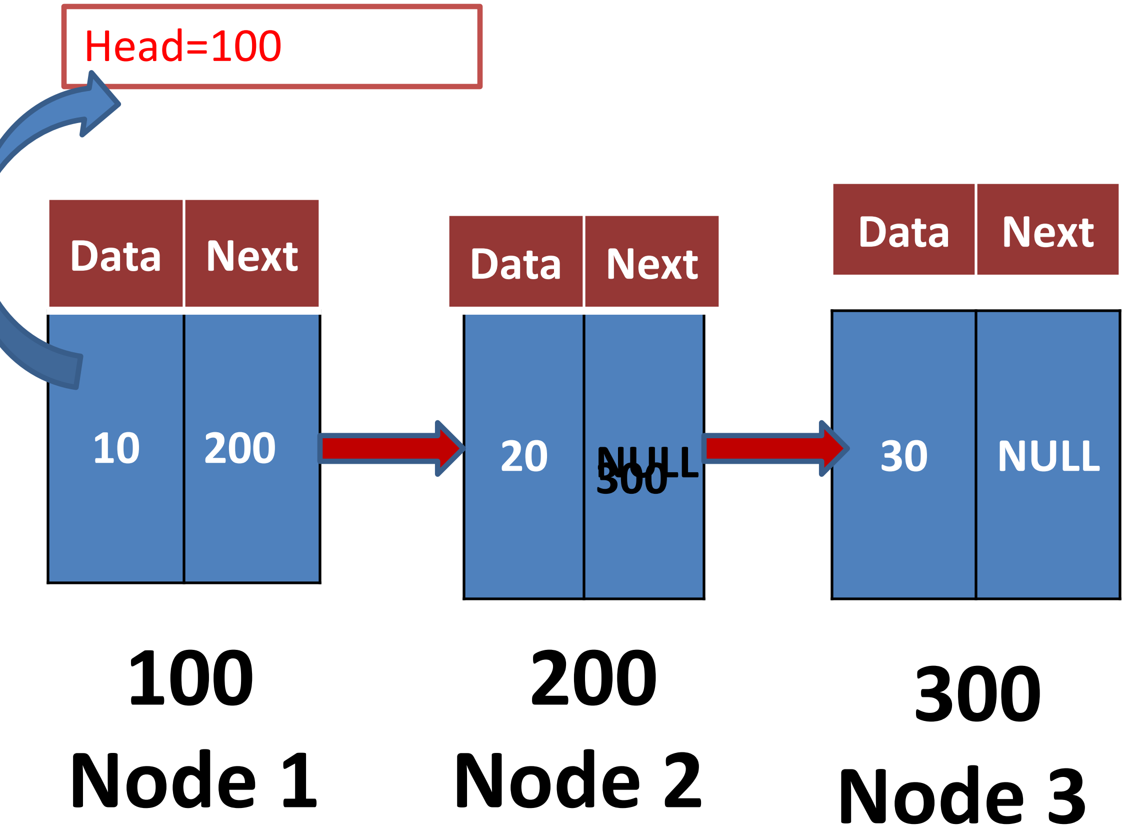
```
temp=head;

While(temp->next!=NULL)
{
temp=temp->next;
}
Temp->next=node2;
```



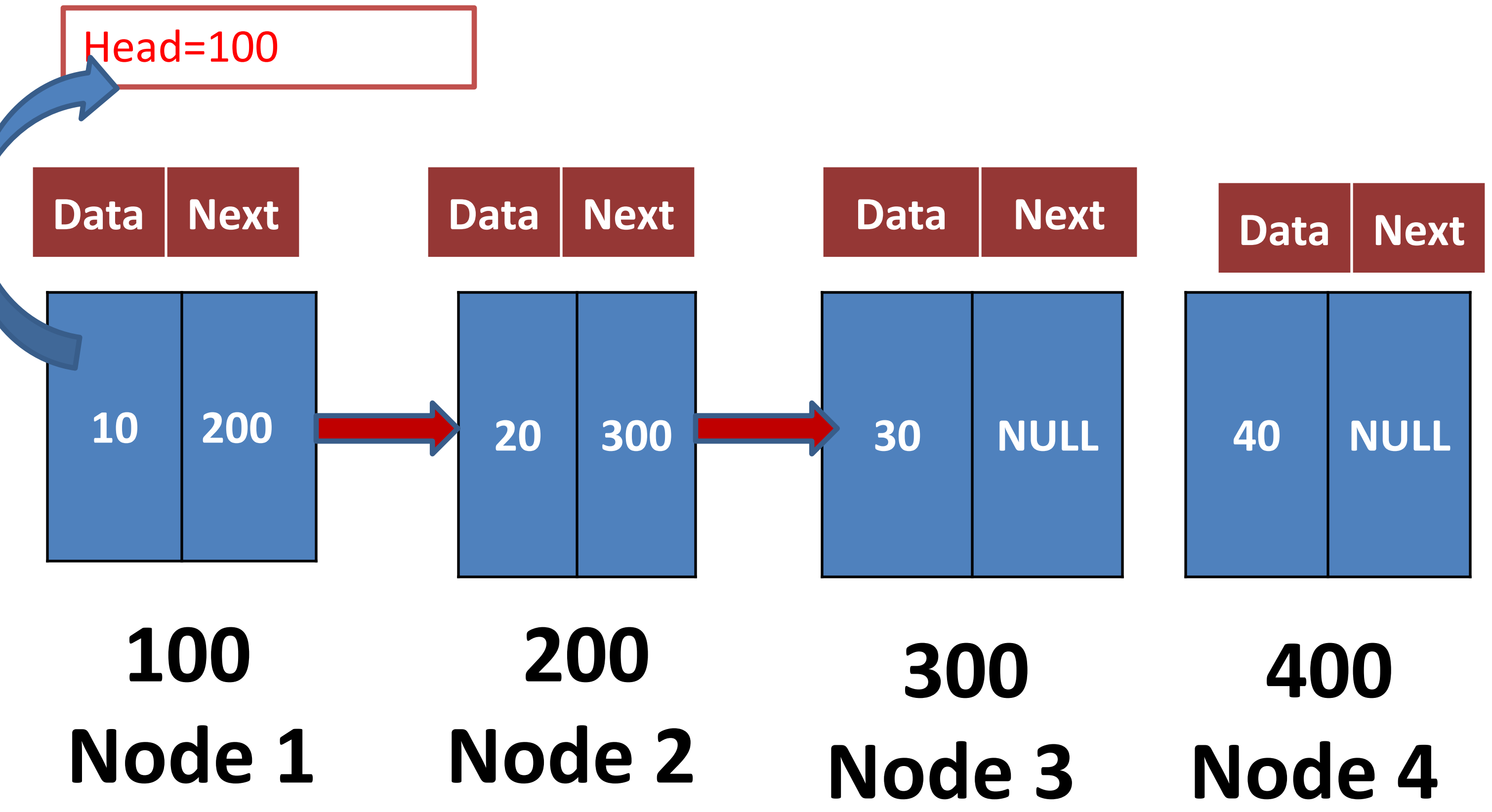
```
Node3=(*struct Node)malloc(sizeof(*struct Node);
```

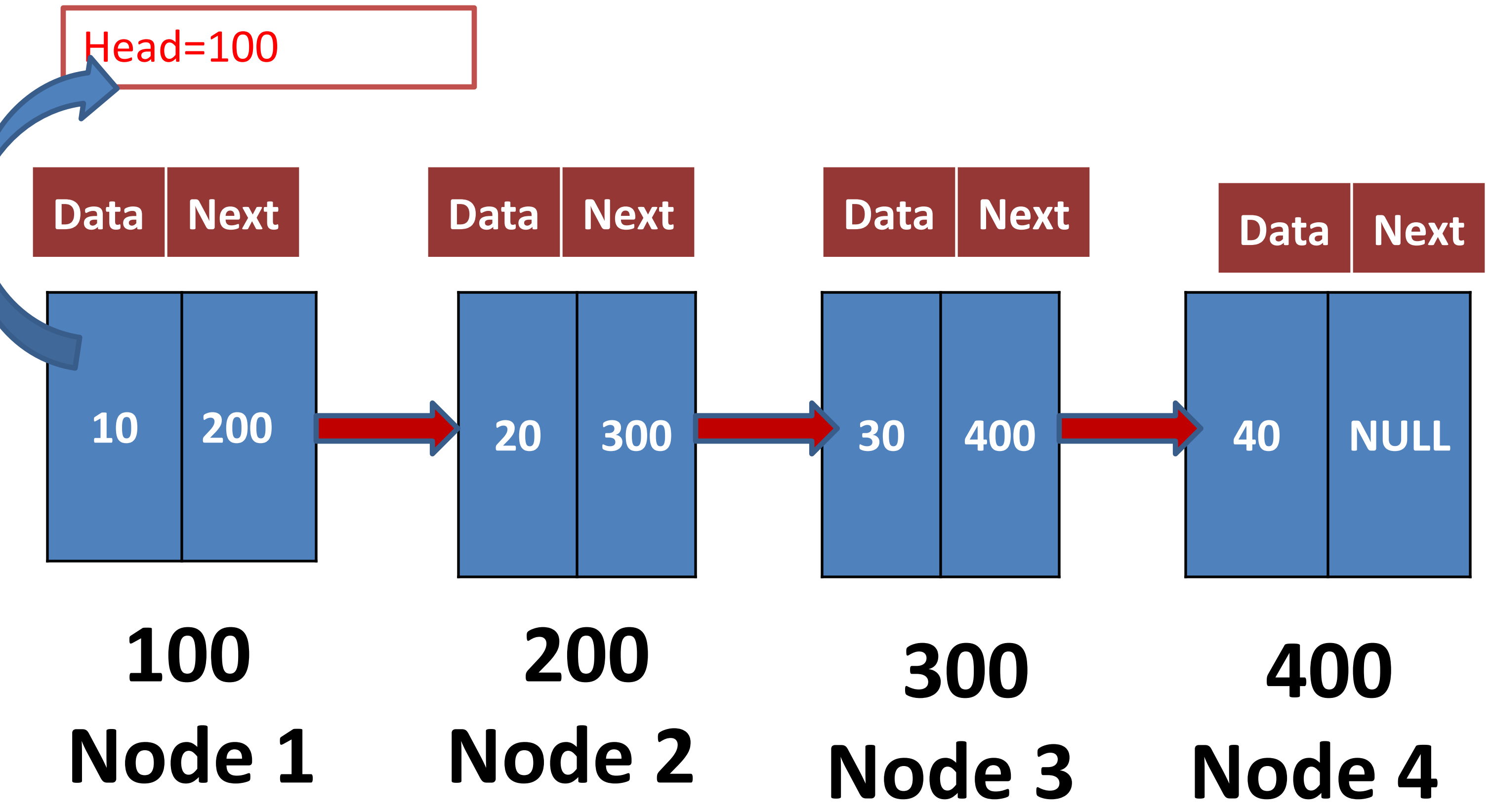
```
Node3->data=30;
Node3->next=NULL;
```

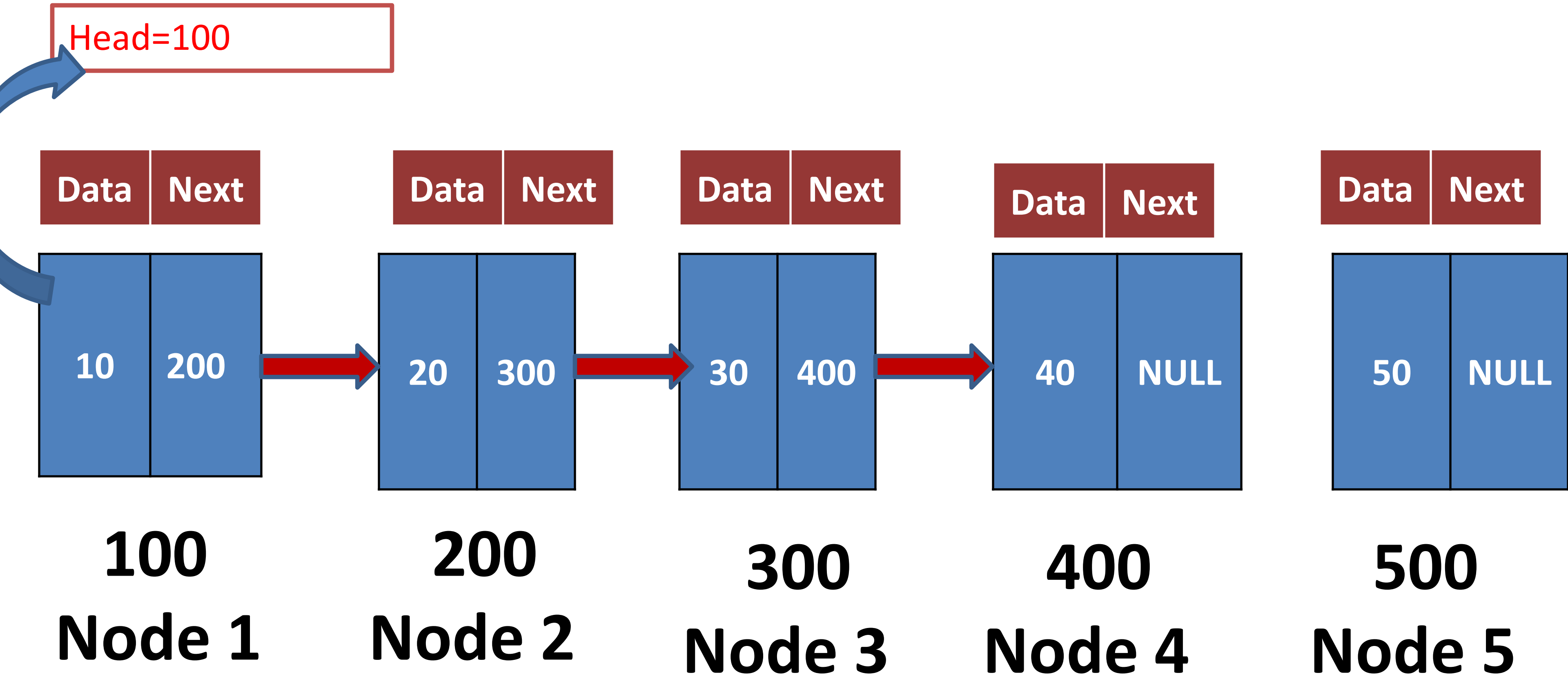
```
If(head==NULL)
{
Head=node1;
}
```

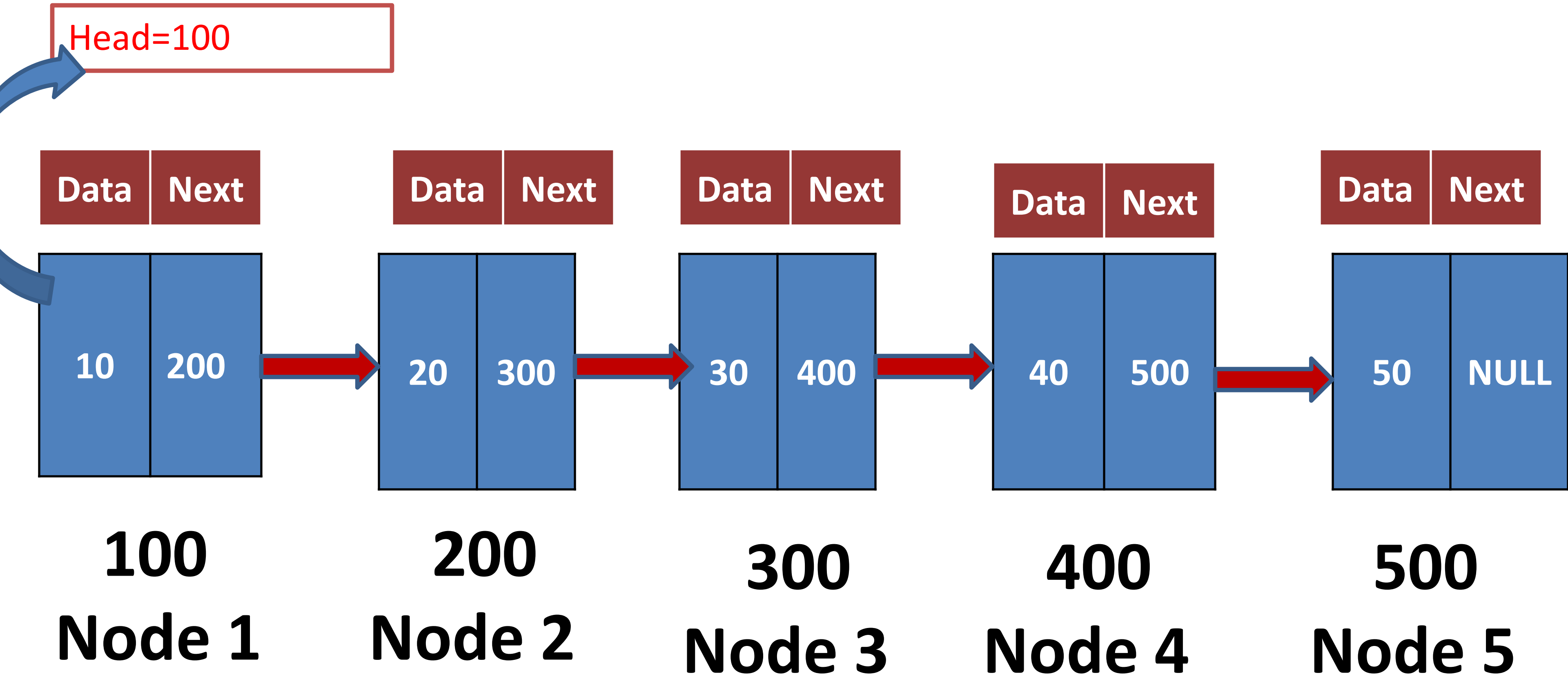


```
temp=head;

While(temp->next!=NULL)
{
temp=temp->next;
}
Temp->next=node3;
```


Head=100
Data
Next
10
200
Data
Next
20
300
Data
Next
30
NULL
Data
Next
40
NULL
100
Node 1
200
Node 2
300
Node 3
400
Node 4


Head=100
Data
Next
10
200
Data
Next
20
300
Data
Next
30
400
Data
Next
40
NULL
100
Node 1
200
Node 2
300
Node 3
400
Node 4


Head=100
Data
Next
10
200
100
Node 1
Data
Next
20
300
200
Node 2
Data
Next
30
400
300
Node 3
Data
Next
40
NULL
400
Node 4
Data
Next
50
NULL
500
Node 5


Head=100
Data
Next
10
200
100
Node 1
Data
Next
20
300
200
Node 2
Data
Next
30
400
300
Node 3
Data
Next
40
500
400
Node 4
Data
Next
50
NULL
500
Node 5

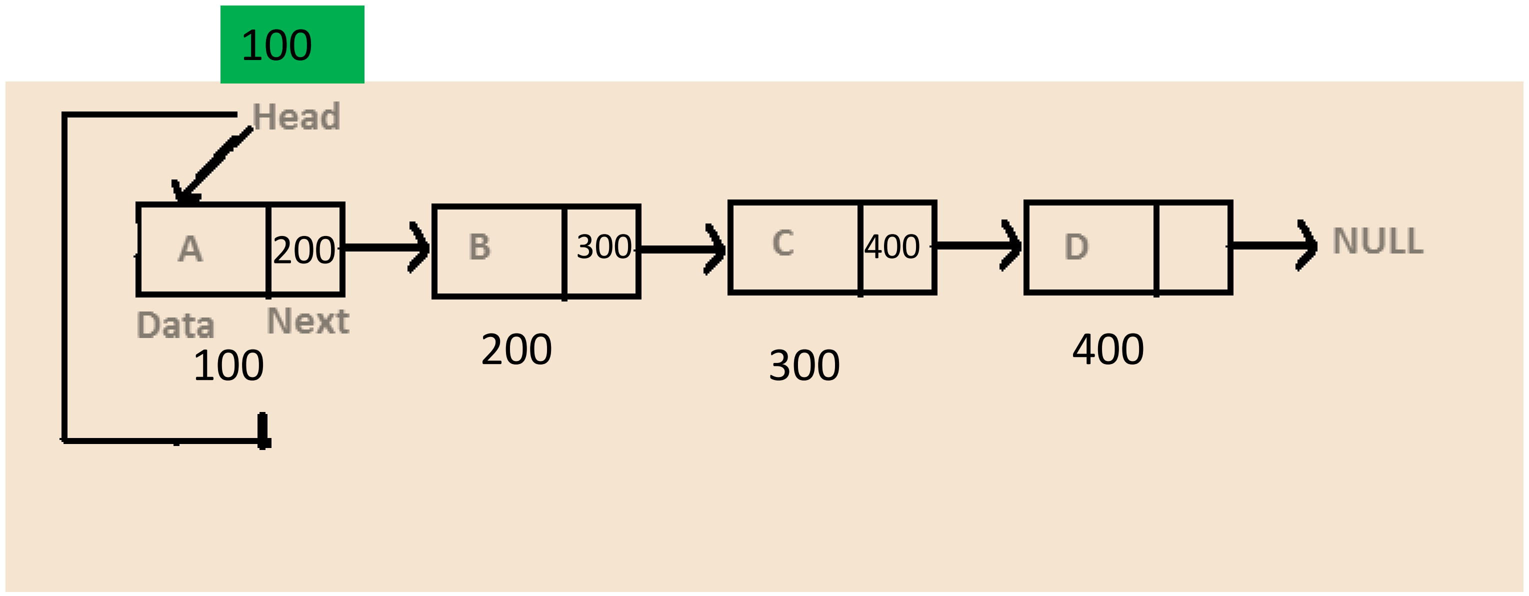
Operations on SLL.

- **Create():** It is used to create the node.
- **Insertion():** it is used to insert the node at
  1. start
  2. Middle
  3. End
- **Deletion():** it is used to delete the node at
  1. start
  2. Middle
  3. End

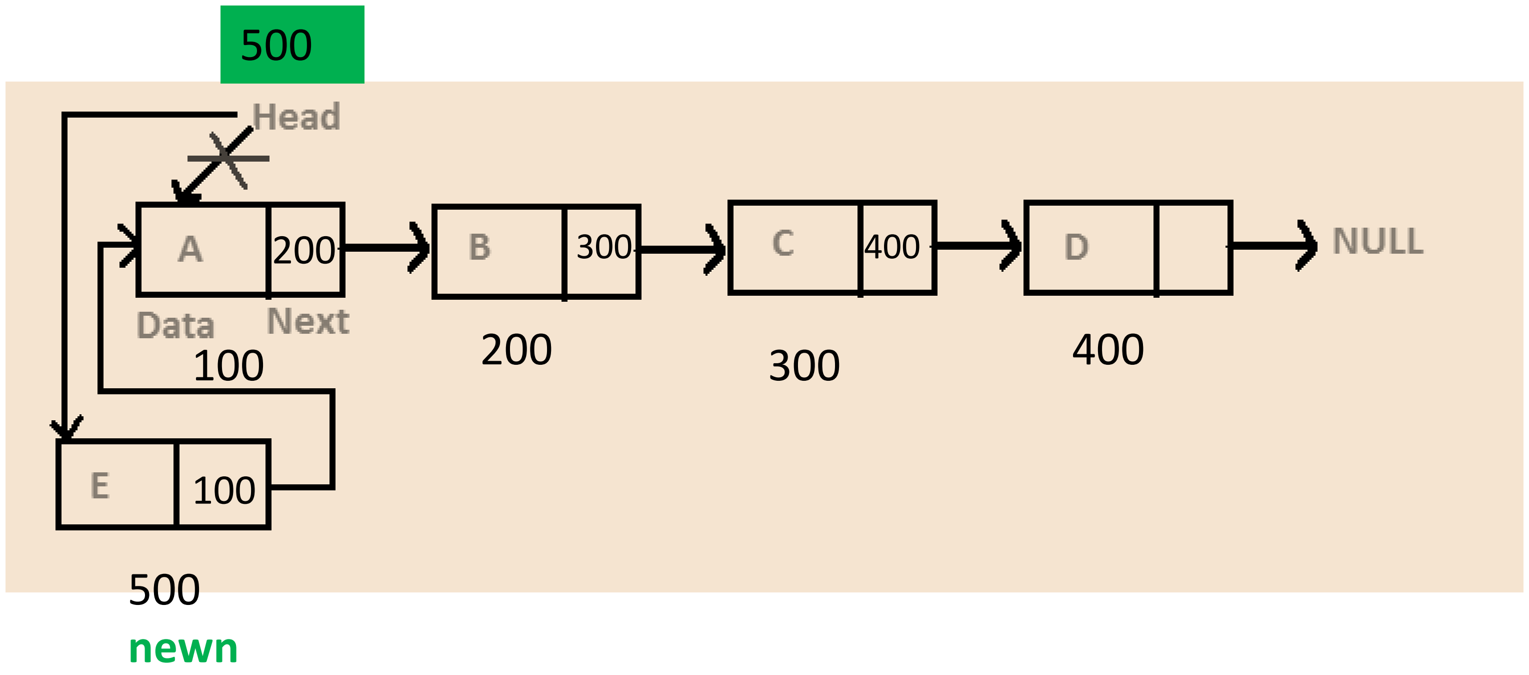
Operations on SLL

- **Display():** it is used to display the nodes in the SLL
- **Search():** it used to search particular node.

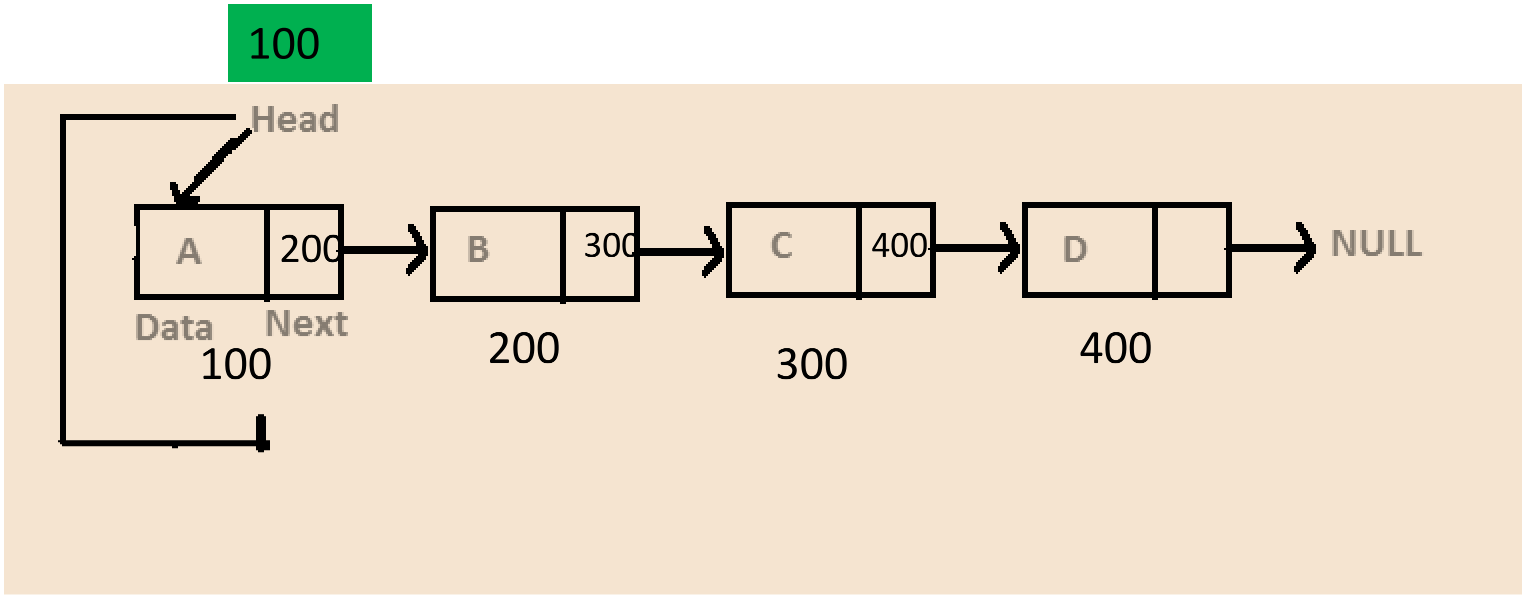
Inserting node at start



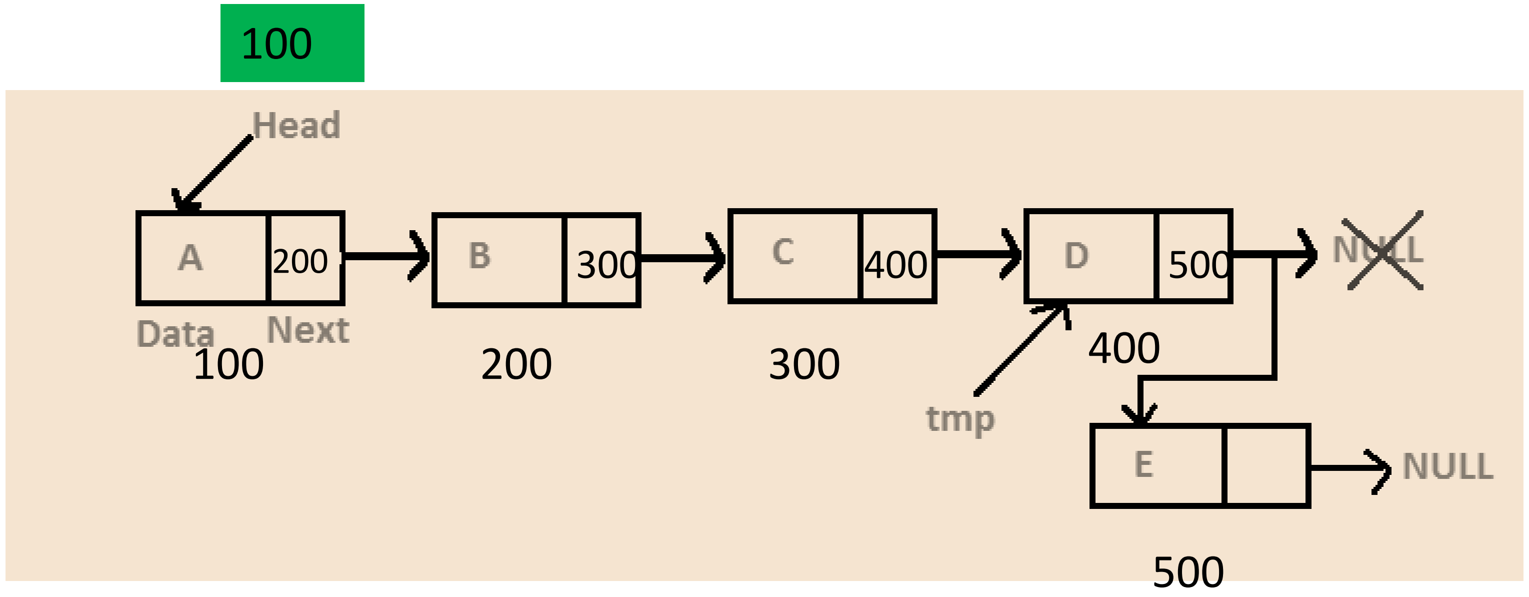
Inserting node at start



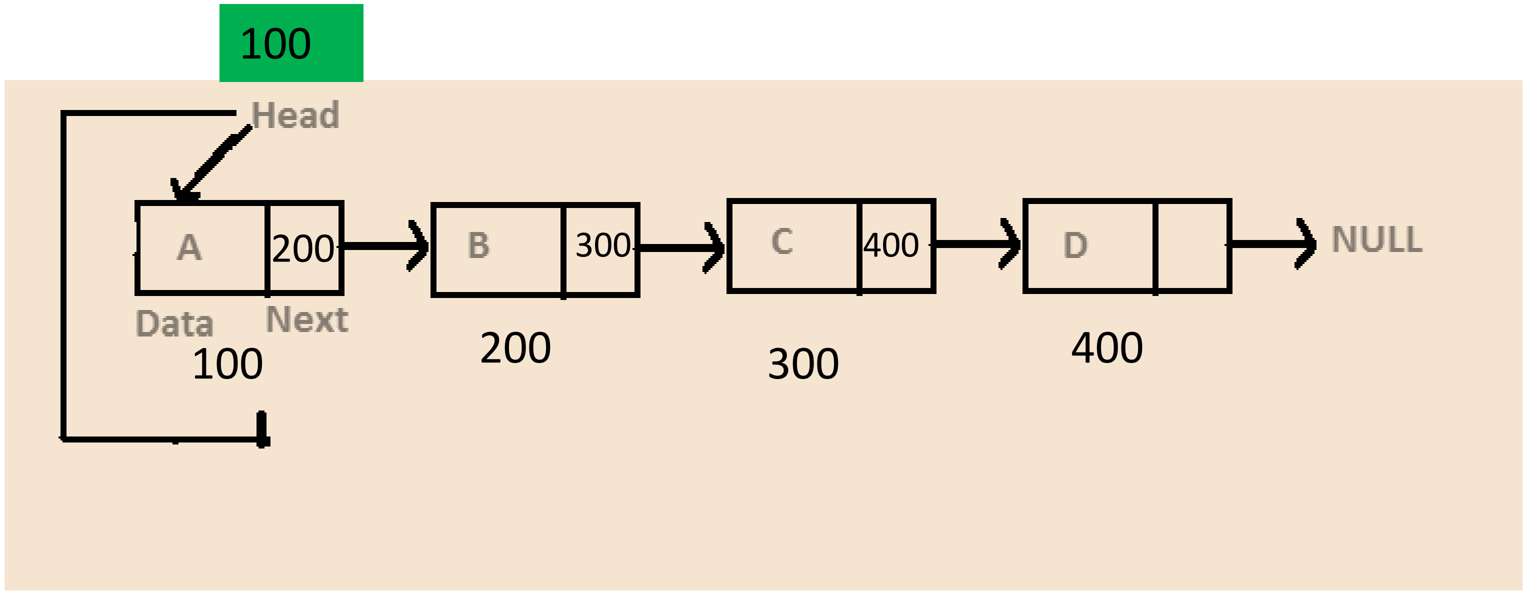
Inserting node at End



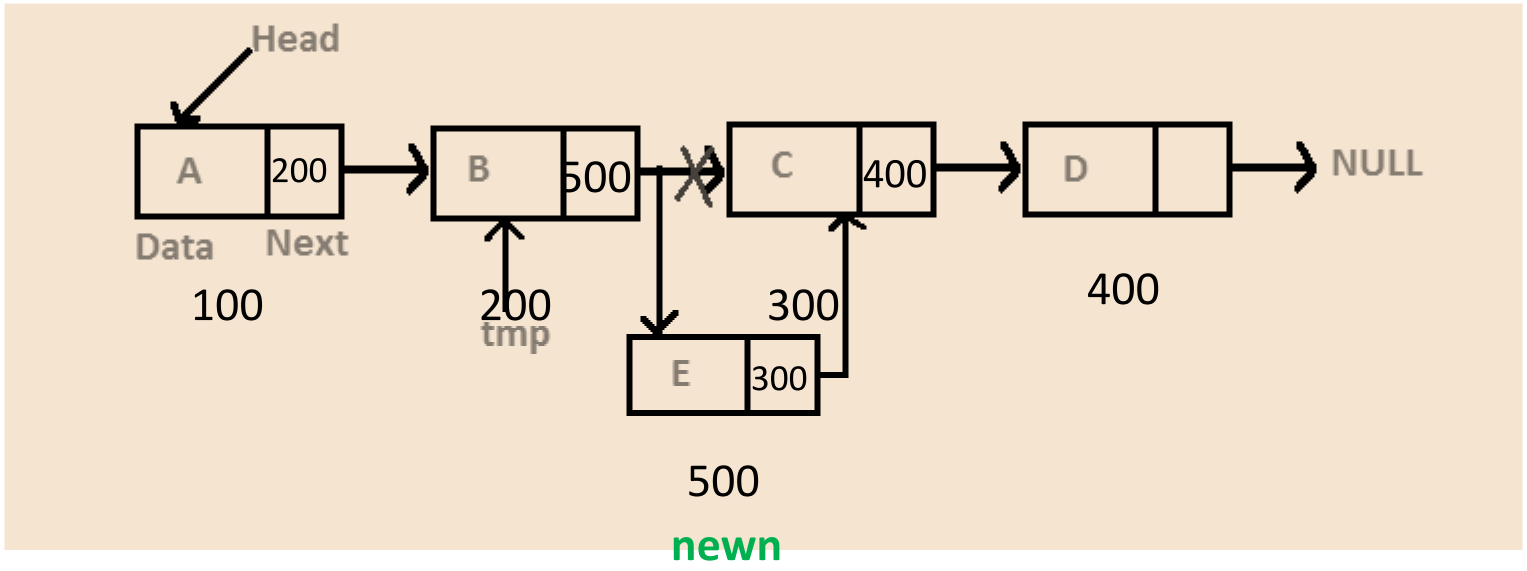
Inserting node at End



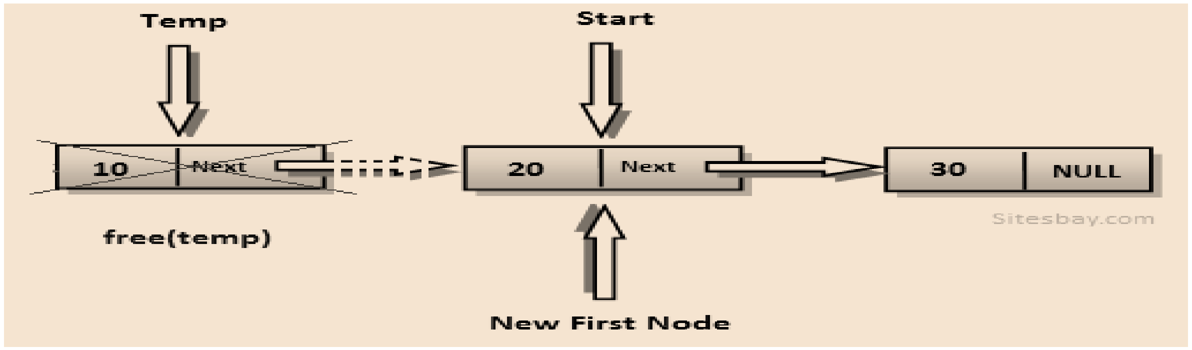
Inserting node at Middle



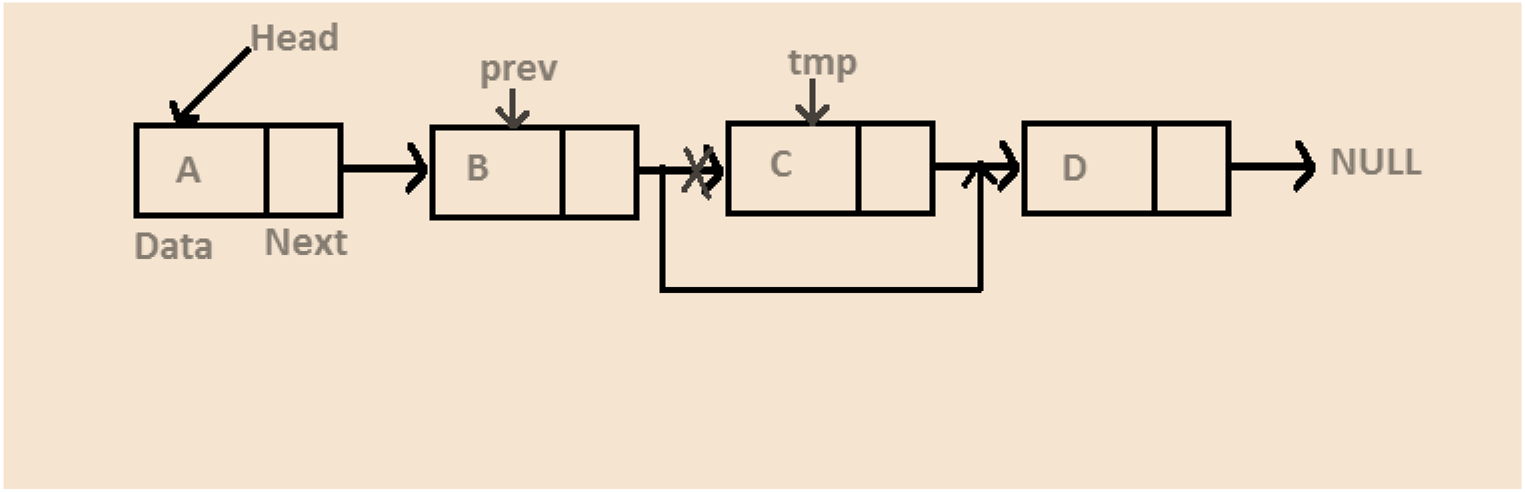
Inserting node at Middle



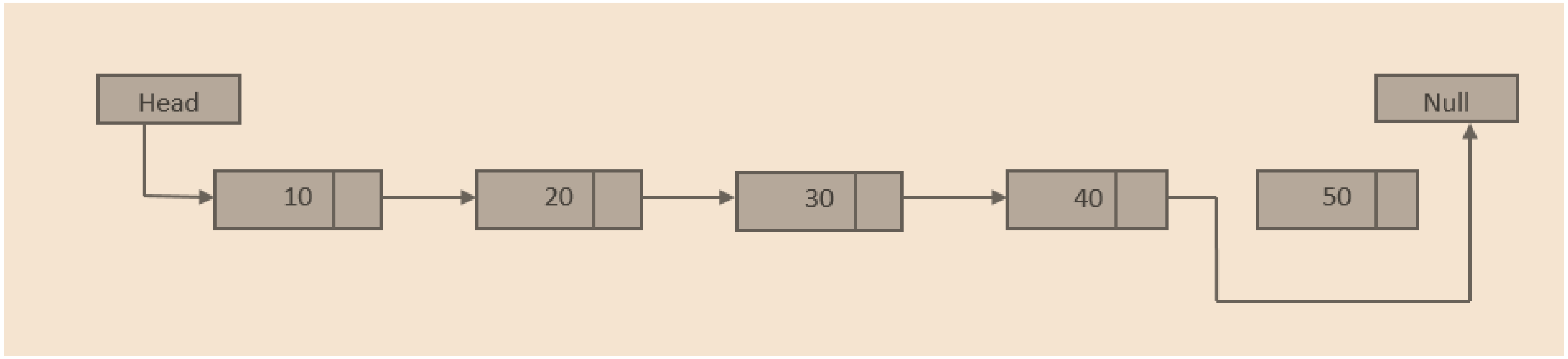
Delete node at start



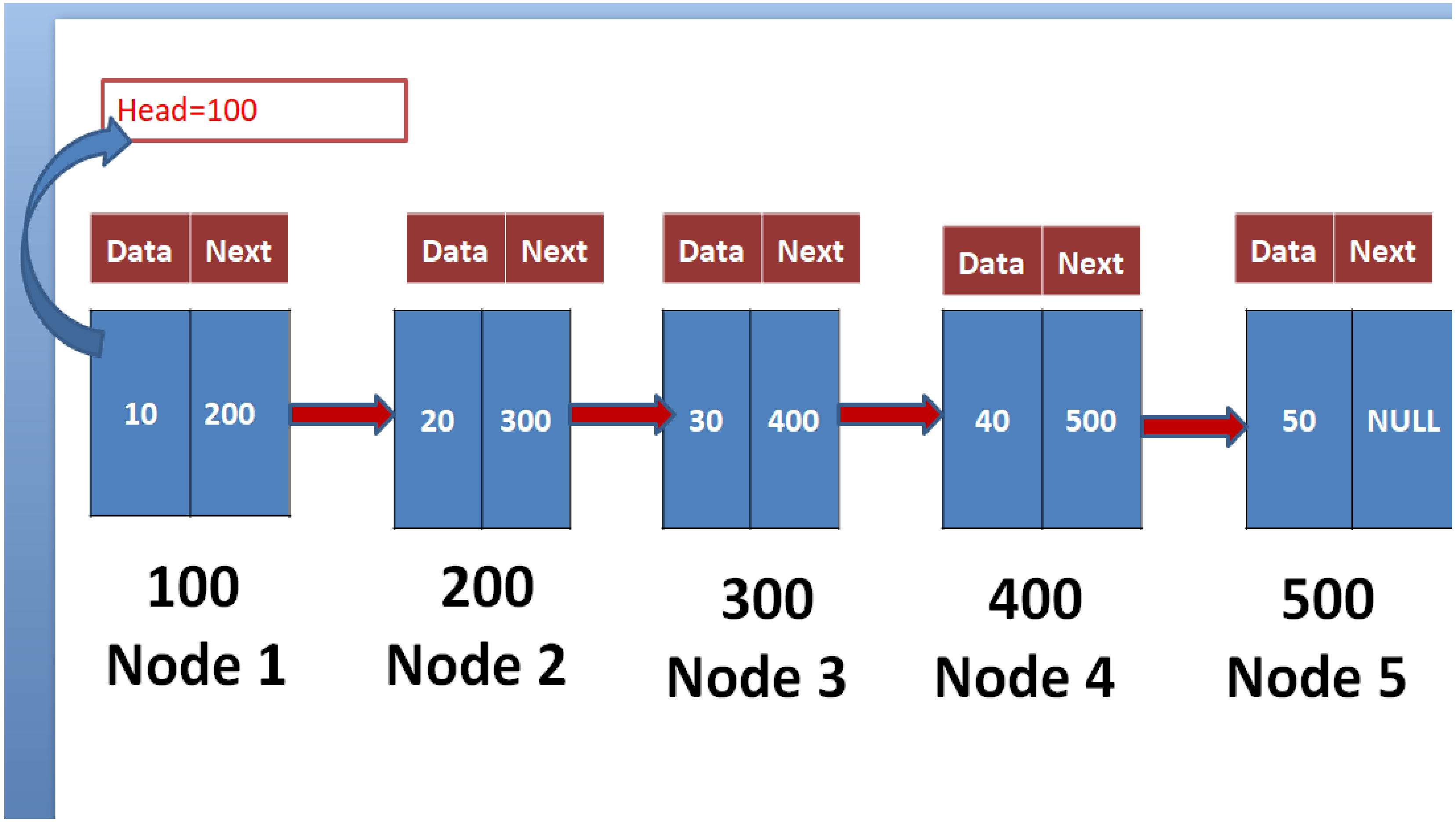
Delete node at middle



Delete node at End

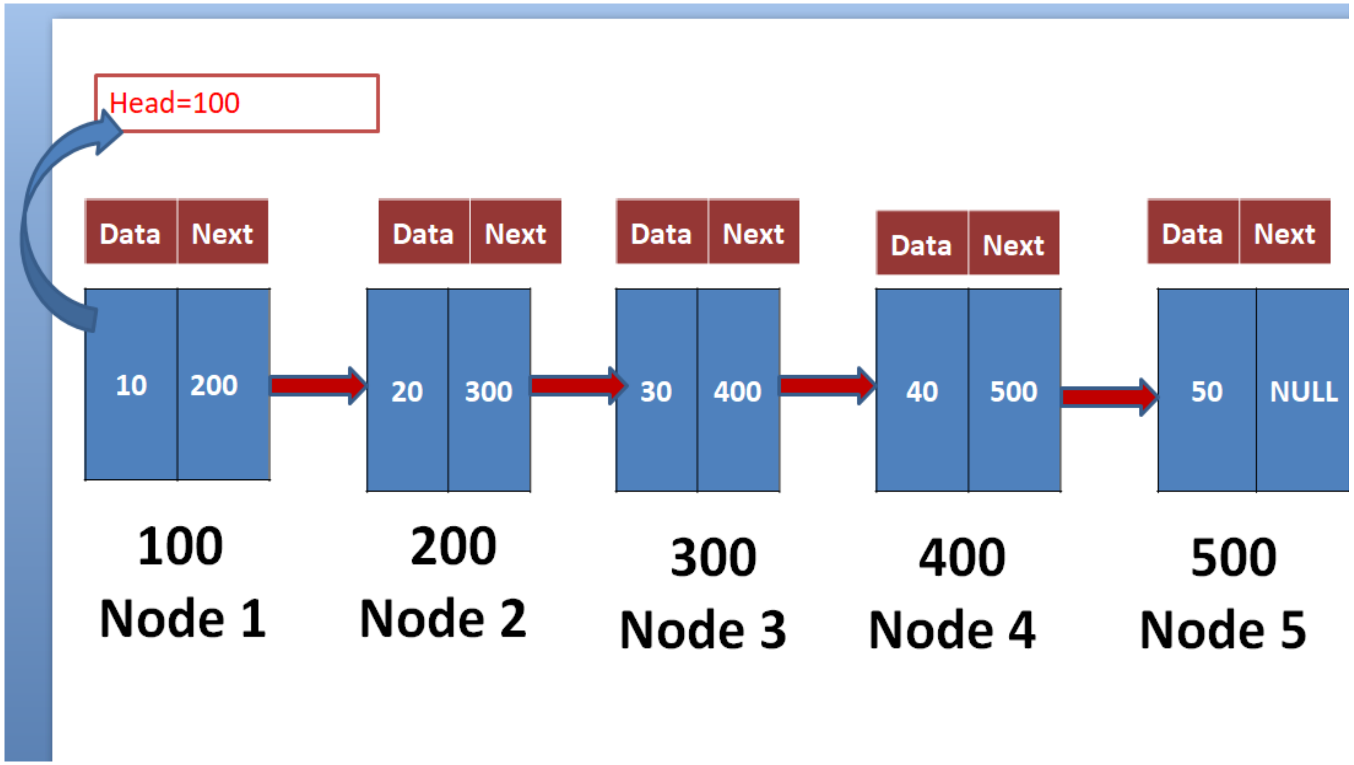


Display


Head=100
Data
Next
10
200
100
Node 1
Data
Next
20
300
200
Node 2
Data
Next
30
400
300
Node 3
Data
Next
40
500
400
Node 4
Data
Next
50
NULL
500
Node 5

Search

- Search a particular node in the list.



Thank You