27/08/2024

PAGE No.
DATE / /

Day 10 of DSA:

Tasks

Check box	Linked List.
☑	Singly Linked List : Basic operations (insertion, deletion)
☑	Doubly Linked List : Implementing doubly linked list operations.
☑	Implement basic operations of singly linked list.
☑	Implement doubly linked list operations.

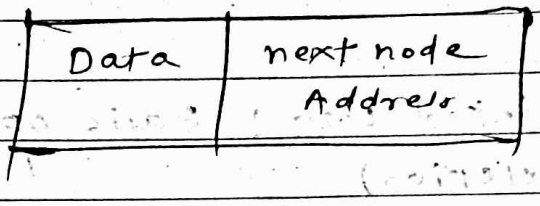
PAGE No.
DATE / /

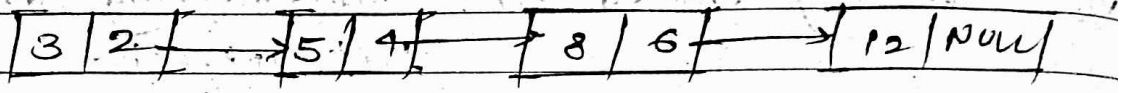
* Linked List :-

Type of linear data structure

↓

Collection of nodes

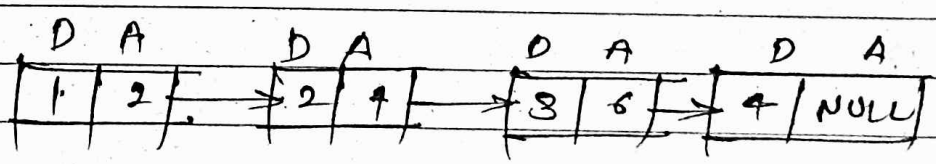




* Types of Linked List :-

1. Singly Linked List.
2. Doubly Linked List.
3. Circular Linked List.
4. Circular doubly Linked List.

* Singly Linked List

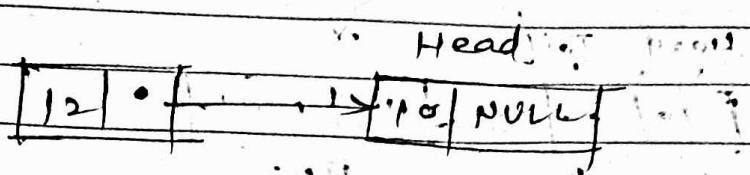


✓ Implementation :-

```
class Node{
    public:
    int data;
    Node* next;
    Node(int data){
        this->data = data;
        this.next = NULL;
    }
```

PAGE No.
DATE / /

* Insert At Head :-



temp node

① temp → next = head;

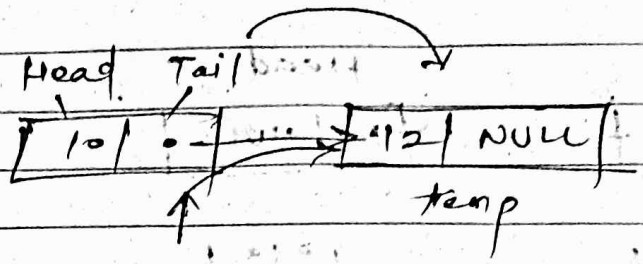
② head = temp

* void insertAtHead(node * &head, int d)
{
node * temp = new Node(d);
temp → next = head;
head = temp;
}

* void print (node * &head) {
Node * temp = Head;
while (temp != NULL)
{
temp → data
temp = temp → next;
}

PAGE No.
DATE / /

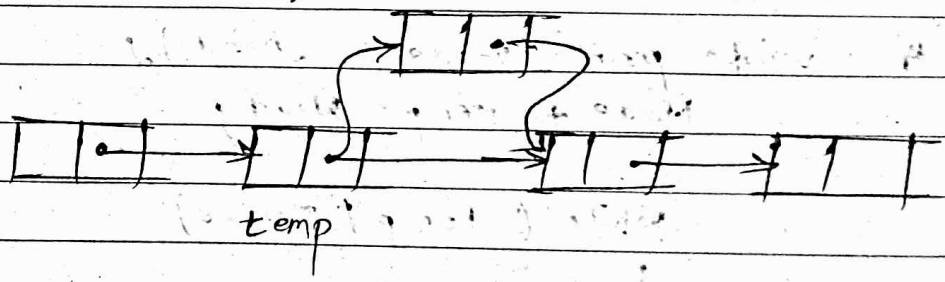
* InsertAtTail :-



tail → next = temp;
tail = temp;

```
void insertAtTail(Node* &tail, int d)
{
    Node* temp = new Node(d);
    tail->next = temp;
    tail = temp;
}
```

* InsertAtMiddle :- NodetoInsert -



NodetoInsert → next = temp → next;
temp → next = NodetoInsert;

Page No.
DATE 11

void insertAtPosition(Node* &head, int position,
int d)

{

Node* temp = head;
int cnt = 1;
while(~~temp != NULL~~ cnt < position-1)
{
temp = temp->next;
temp को उसके आगे element से
जोड़ दो।
cnt++;
}

Node* NodeToInsert = new Node(d);

NodeToInsert->next = temp->next;
temp->next = NodeToInsert;

}

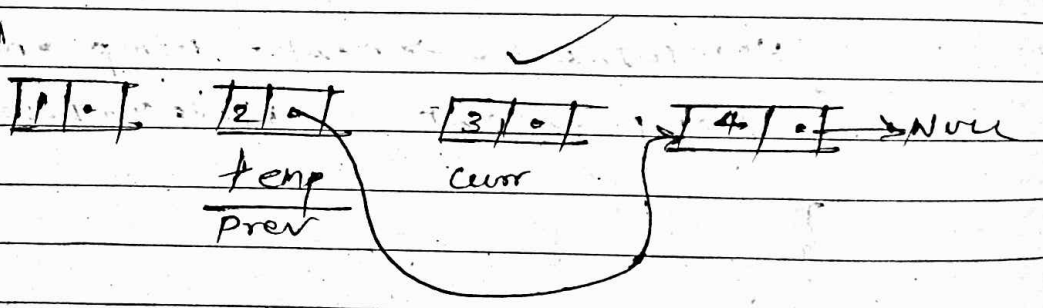
PAGE No.
DATE / /

Delete Node

Destructor in class Node

```
~Node()
{
    int value = this->data;
    if(this->next != NULL)
    {
        delete next;
        this->next = NULL
    }
}
```

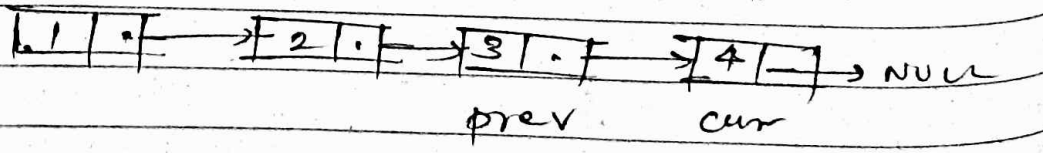
(1) Middle Node



temp->next = curr->next

delete curr;

(2) Last Node.

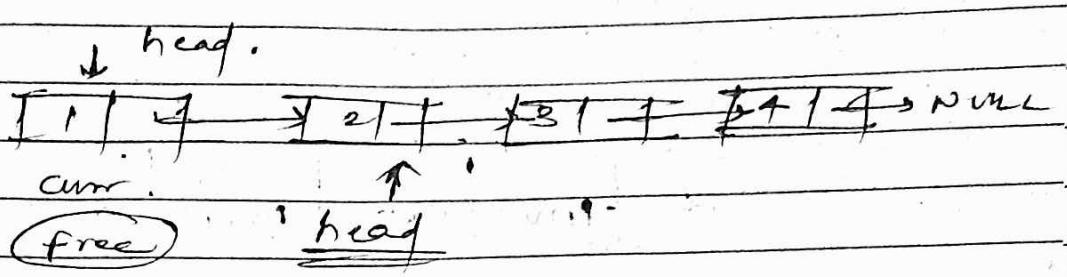


prev->next = curr->next.

delete curr;

PAGE No.
DATE / /

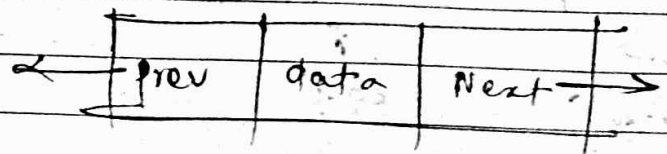
(3) First Node →

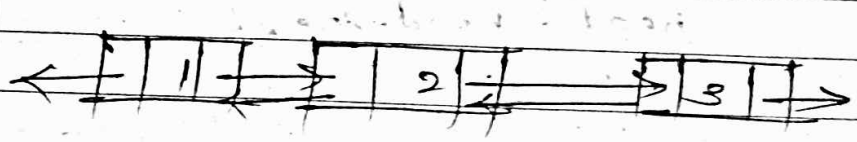


head = head->next

PAGE No.
DATE / /

* Doubly Linked List:-





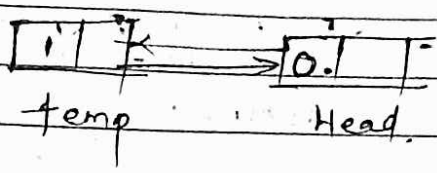
* Implementation:-

```
class Node
{
    public:
    int data;
    Node* prev;
    Node* next;

    Node(int d)
    {
        this -> data = d;
        this -> prev = NULL;
        this -> next = NULL;
    }
};
```

PAGE No.
DATE / /

* Insert At head :-



(1) Create temp Node;

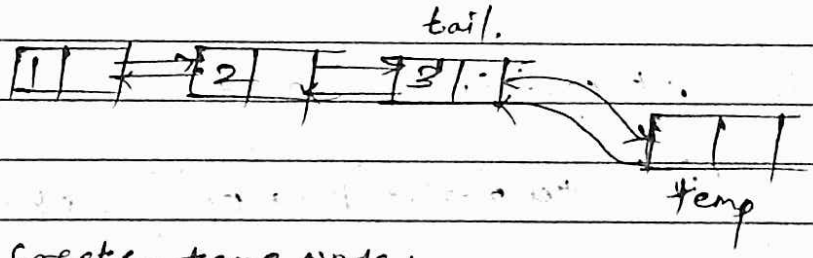
Node * temp = new Node(d);

(2) temp -> next = head.

(3) head -> prev = temp

(4) head = temp;

* Insert At Tail :-



(1) Create temp Node;

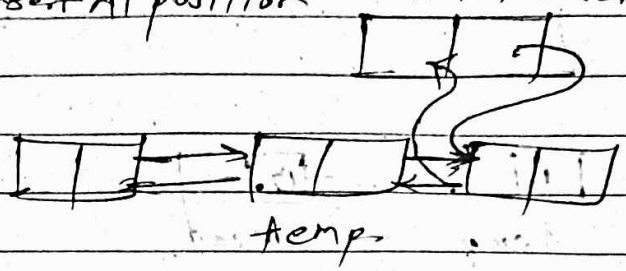
(2) tail -> next = temp;

(3) temp -> prev = tail;

(4) tail = temp;

PAGE No.
DATE / /

* Insert At position



(1) Create Node

(2) NodeToInsert -> next = temp -> next;

(3) temp->next->prev = NodeToInsert

(4) temp -> next = NodeToInsert

(5) NodeToInsert -> prev = temp;

* Delection :-

At position == head

(1) temp -> next -> prev = NULL;

(2) head = temp -> next;

(3) temp -> next = NULL;

(4) Memory free.