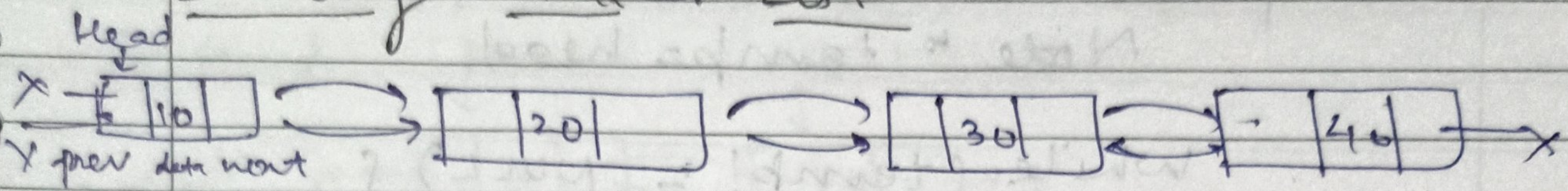
LINKED LIST CLASS - 2

PAGE No
DATE

Doubly Linked List




```
class Node
{
    int data;
    Node * prev;
    Node * next;
}
```

Node creation

Code

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

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

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

```
void print (Node * &head) {
    Node * temp = head

    while (temp != NULL) {
        cout<< temp -> data << " ";
        temp = temp -> next ;
    }
}
```

```
int getLength (Node * head) {
    int len = 0 ;
    Node * temp = head ;

    while (temp != NULL) {
        temp = temp -> next ;
        len ++ ;
    }
    return len ;
}
```

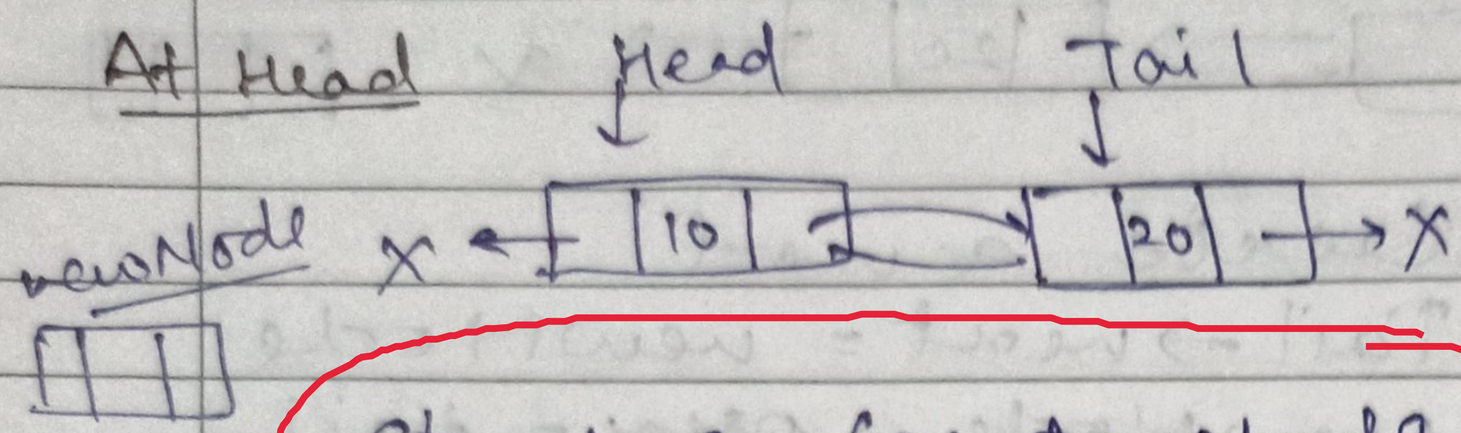
```
int main() {
    Node* first = newNode( 10);
    Node * second = newNode(20);
    Node * third = newNode(30);

    first -> next = second;
    second -> prev = first ;

    second -> next = third;
    third -> prev = second;
    print(first);
}
```

Insertion

At Head



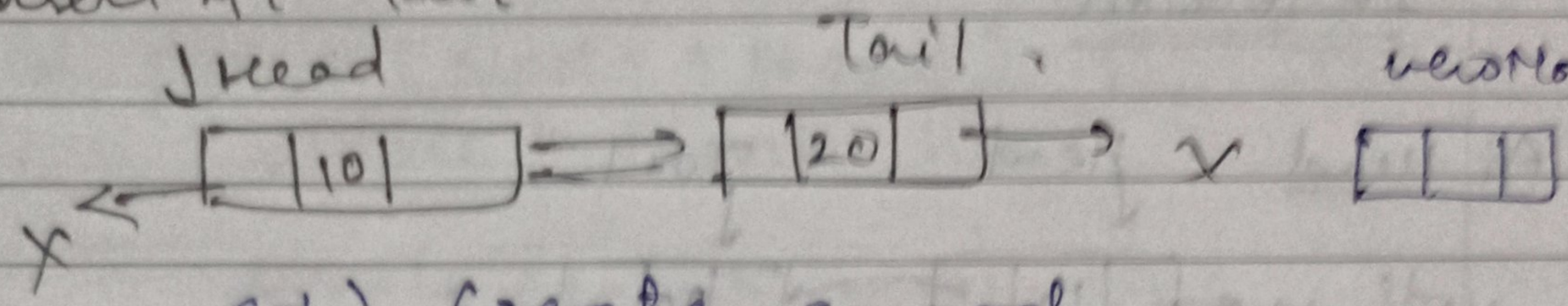
Step-1 → Create Node
Step 2 → newNode → next = head
step 3 → head → prev = newNode
step 4 → Head = newNode

```
void InserAtHead(Node * &head, Nod^
                 &tail, int data){

    if(head == NULL){
        Node* newNode(~~dat~~ = new Node(data);
        head = newNode;
        tail = newNode;
    }
    else {
        {
        Node * newNode = newNode.(data);
    newNode → next = head;
     head → prev = newNode;
      head = newNode
}
}
```

Yaha Hum head or tail via reference pss karenge ki hum chahte hai ki hamare original value me updation ho

Insert At Tail



(-1) Create a node
(-2) Tail → next = newNode
(-3) NewNode → prev = tail
(-4) tail = NewNode

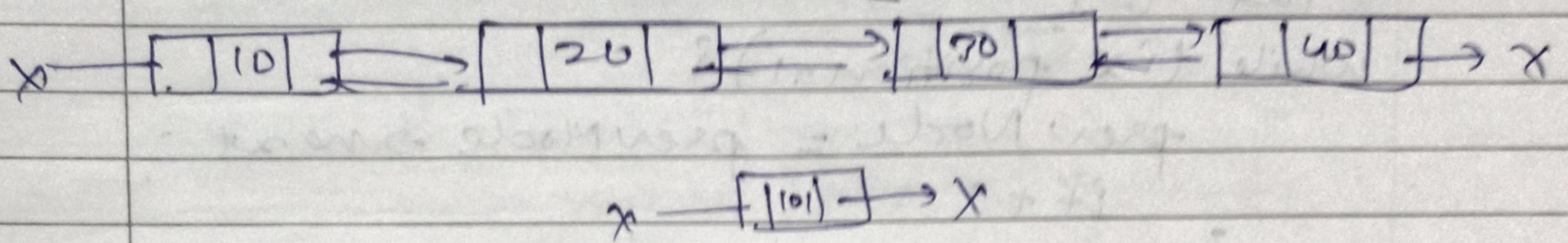
Code

```
void InsertAtTail(Node *&head, Node
        *&tail, int data){

    if(head == NULL){
        Node *newNode = new Node
                                (data);
        head = newNode;
        tail = newNode;
    }
    else {

        Node *newNode = new Node(data)
        tail->next = newNode;
        newNode->prev = tail;
        tail = NewNode;
    }
}
```

Insert At Position



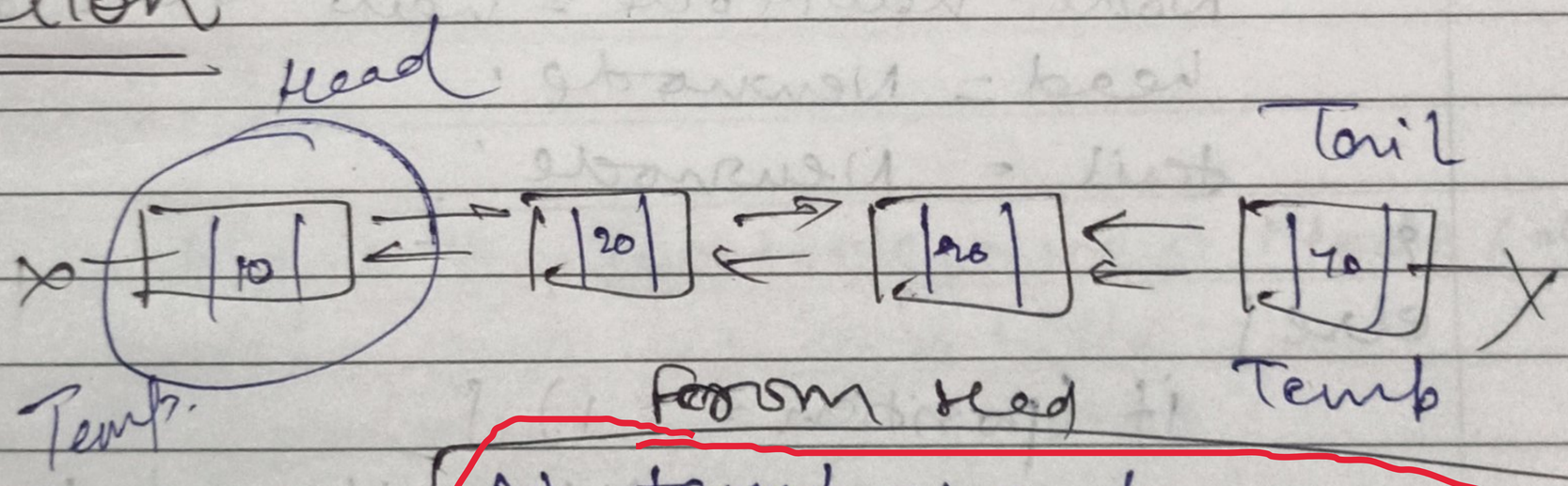
S-1) find prev & curr.
S-2) create a node
S-3) prevNode → next = NewNode
S-4) NewNode → prev = prevNode.
S-5) curr → prev = newNode
S-6) NewNode → next = curr;

Code

```
Void insertAtPos (Node *head, Node * tail,
int position, int data) {
    if (head == NULL) {
    Node * newNode = new Node (data);
    head = Newnode;
    tail = Newnode;
}
else {
    if (position == 1) {
        insertAtHead (head, tail, data);
        return;
    }
int len = getLength(head);
    if (position >= len) {
        insertAtTail (head, tail, data);
        return;
}
```

```
Node * prevNode = head;
int i = 1;
while (i < position-1) {
    prevNode = prevNode -> next;
    i++;
}
Node * curr = prevNode;
Node * newNode = new Node(data);
prevNode -> next = newNode;
newNode -> prev = prevNode;
curr -> prev = newNode;
NewNode -> next = curr;
}
```

Deletion



From Tail

A) temp = tail
B) tail = tail -> prev
C) temp -> prev = NULL
D) tail -> next = NULL
E) delete temp.

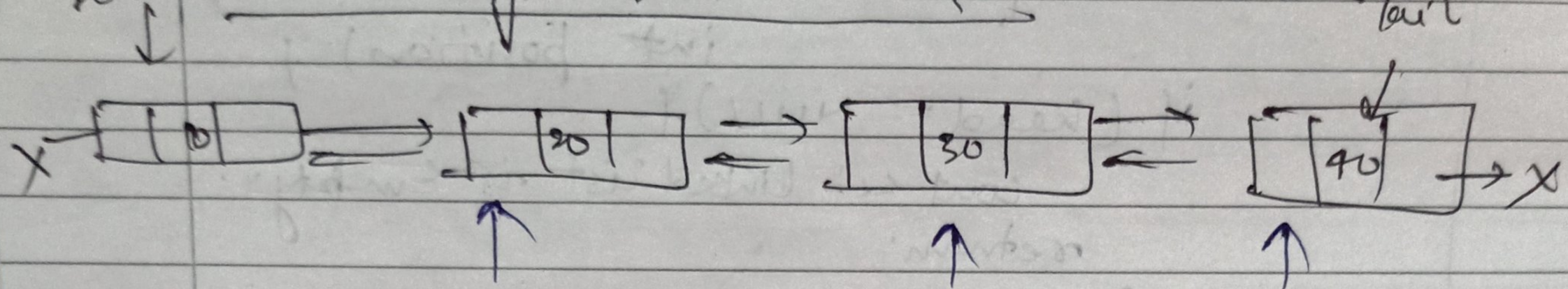
From Head

A) temp = head
B) Head = head -> next
C) Head -> prev = NULL
D) temp -> next = NULL
E) delete temp.

```
Void delete (Node * & head, Node * & tail,
                                int position);
if (head == NULL) {
        cout << " Linked list is Empty ";
    return;
}
if (head -> next == NULL) {
    Node * temp = head;
    head = NULL;
    tail = NULL;
    delete temp;
    return;
}
if (position > getlength) {
    cout << " Invalid "
    return; }
if (position == 1) {
    Node * temp = head;
    head = head -> next;
    head -> prev = NULL;
    temp -> next = NULL;
    delete temp;
    return;
}
int len = getlength (head);
if (position == len) {
    Node * temp = tail;
    tail = tail -> prev;
    temp -> prev = NULL;
    tail -> next = NULL;
    delete temp;
    return;
}
}
```

PAGE No
DATE

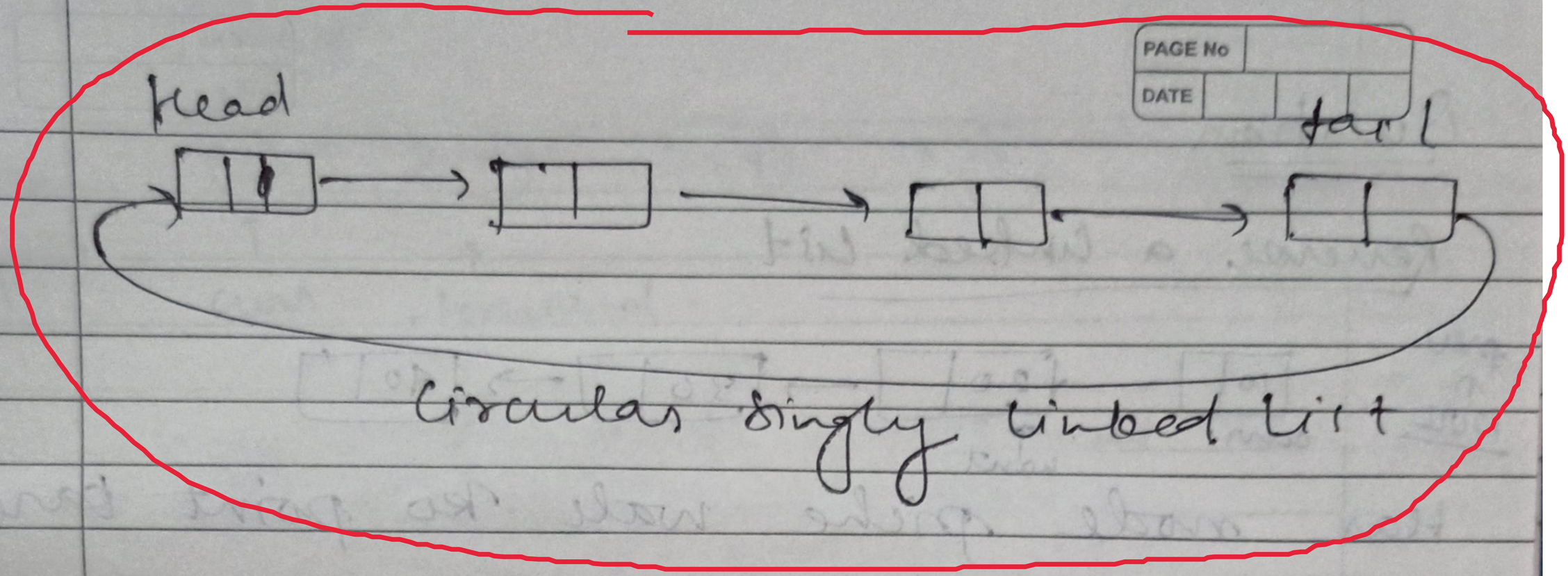
Head // delete from middle

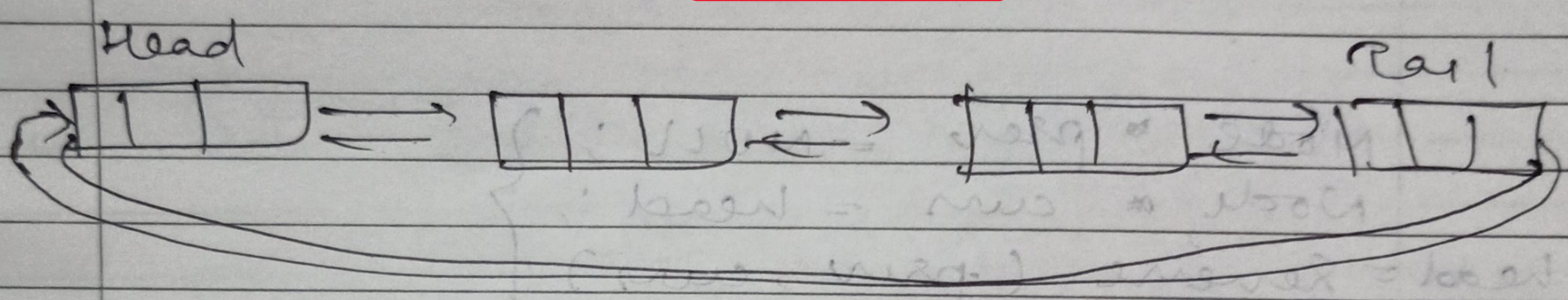


1) find left, curr, right
2) left → next = right
3) right → prev = left
4) curr → prev = NULL
5) curr → next = NULL
6) delete curr;

CODE

```
int i=1;  → Node * left = head;
while (i < position-1){
        left = left → next;
        i++;
}
Node * curr = left → next;
Node * right = curr → next;
left → next = right;
right → prev = left;
curr → next = NULL;
curr → next = NULL;
delete curr;
return;
}
```





Circular Doubly linked list

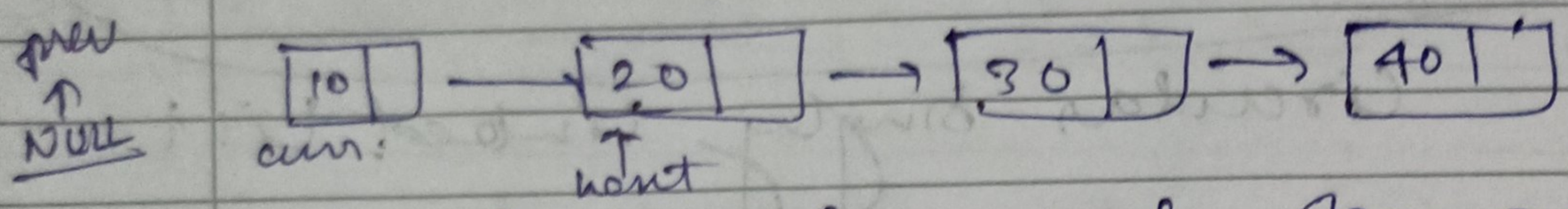
tail → next = head

head → prev = tail.

PAGE No
DATE

Question

Reverse a Linked List

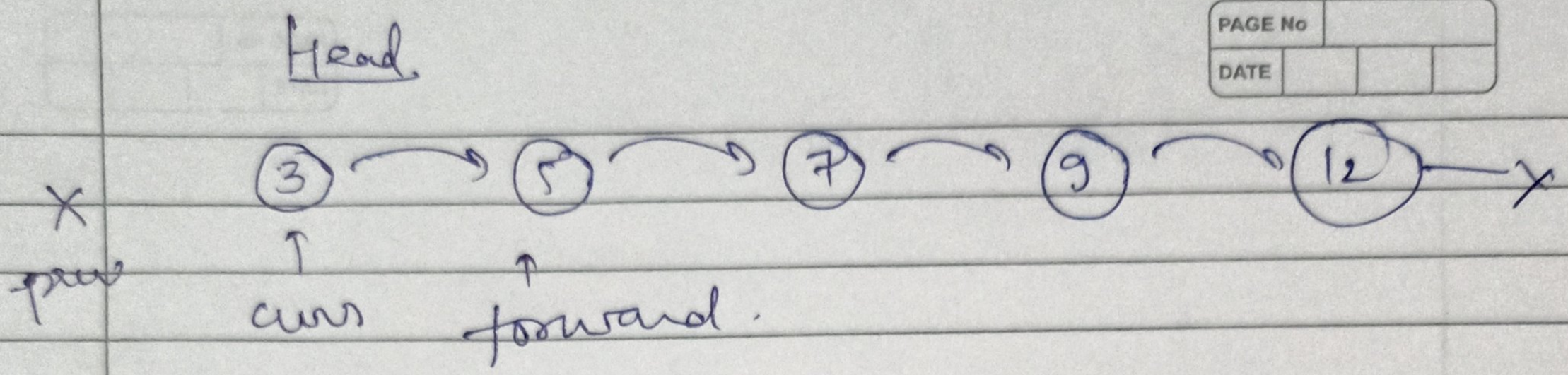


Har node piche wale ko point kare.

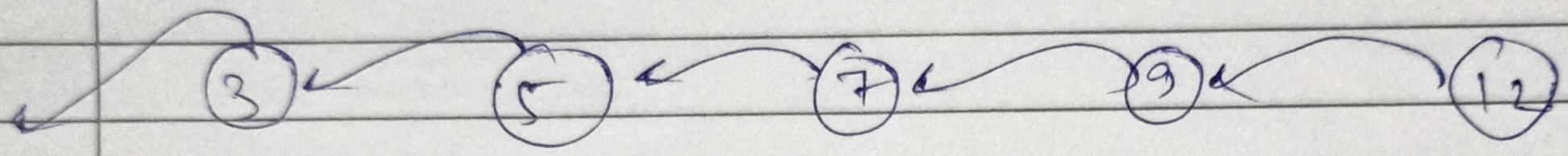
```
Node * prev = NULL;
Node * curr = head;
head = reverse (prev, curr);
```

~~Reverse~~

```
Node* reverse (Node *&prev, Node *&curr) {
    if (curr = NULL) {
        return prev;
    }
    Node * ~~next~~ forward = curr -> next
    curr -> next = prev;
    reverse (curr, forward);
}
```



case forward = curr → next
curr → next = prev.



Loop Code

```
Node * Reverse (Node * head) {
    Node * prev = NULL
    Node * curr = head;

    while (curr != NULL) {
        Node * temp = curr → next
        curr → next = prev;
        prev = curr;
        curr = temp;
    }
    return prev;
}
```