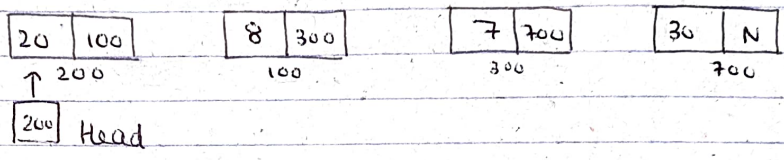
Date: / /
Page:

Day - 114

Linked List - 2

* **Delete a Node:**



⇒ **Delete First Node**

⇒ We simply have to move the Head pointer to the next node.

⇒ Also, after that we have to release the memory of that node.

```
if( Head != NULL){
    Node * temp = Head;
    Head = Head → next;
    delete temp;
}
```

⇒ Sometimes, it also possible that Head pointing to NULL. In that case, we don't do anything.

⇒ **Delete Last Node**

⇒ Create a curr pointer and traverse it to the last node.

⇒

```
Node *curr= Head;
while( curr → next != NULL){
    curr = curr → next;
}
```

Date: / /
Page:

⇉ Here, we will encounter a problem that we can't make the last node next part NULL because this have to null.

⇉ For this, we have to require another pointer 'prev'.

So,

⇉
```
Node *curr = Head;
Node *prev = NULL;
while(curr->next != NULL){
    prev = curr;
    curr = curr->next;
}
delete curr;
prev->next = NULL;
```

⇉ Now, we have to check for edge cases, that are —

⇉ LL doesn't exist.

⇉ LL has only 1 node.

```
if(Head->next == NULL){
    Node *temp = Head;
    Head = NULL;
    delete temp;
}
```

⇉ So, we have covered all the edge cases.

Date : / /
Page:

* **Delete a particular Node:**

⇒ First create a 'curr' ~~position~~ pointer and move it to X-1 times.

⇒ Also, we require a 'prev' pointer again.

⇒
```
Node *curr = Head;
Node *prev = NULL;
x--;
while(x--){
    prev = curr;
    curr = curr->next;
}
prev->next = curr->next;
delete curr;
```

⇒ If the value of X is 1 then above code don't workes.

⇒ So,

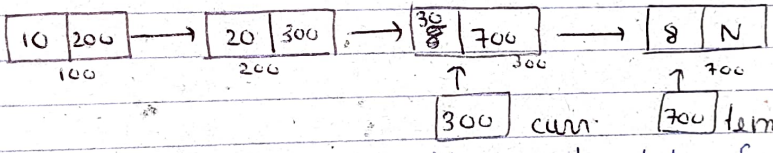
```
if(x==1){
  Node *temp = Head;
  Head = Head->next;
  delete temp;
}
```

⇒ Now, if we want to do this by using recursion.

```
Node* deleteNode(Node *curr, int x){
    if(x==1){
        Node *temp = curr->next;
        delete curr;
        return temp;
    }
    curr->next = deleteNode(curr->next, x-1);
    return curr;
}
```

Q. We have a LL and we have to delete a Node that pointed by curr. But we don't have Head pointer.

⇒



⇒ So, we can do that we will copy the data of next node to the node pointed by curr.

⇒ Create a temp pointer.

```
Node *temp = curr->next;
curr->data = temp->data;
curr->next = temp->next;
delete temp;
```