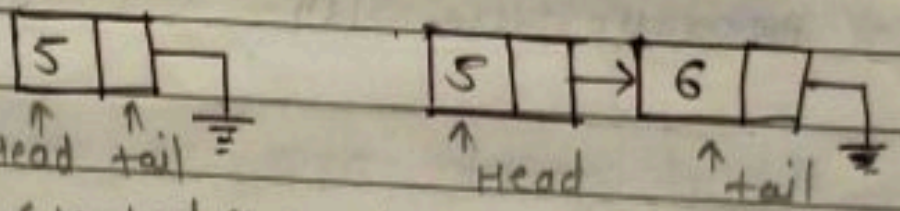
Assignment No:- 9

Que. Explain operations like creation, insertion, deletion, searching, sorting on SLL with example & pictorial representation:-

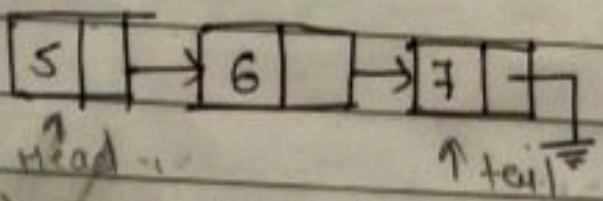
→

1) Creation:-

- Let we have to create 3 nodes, Here dynamic memory is allocated.
- first node is created & then it is linked with subsequent nodes.
- 'new' keyword is used to create new node



a) Created first node b) created second node & linked it with 1st node



c) created third node & linked it with second node

pseudocode:

```
nnode = new node;
nnode -> next = NULL;
cout << "enter data";
cin >> nnode -> data;
```

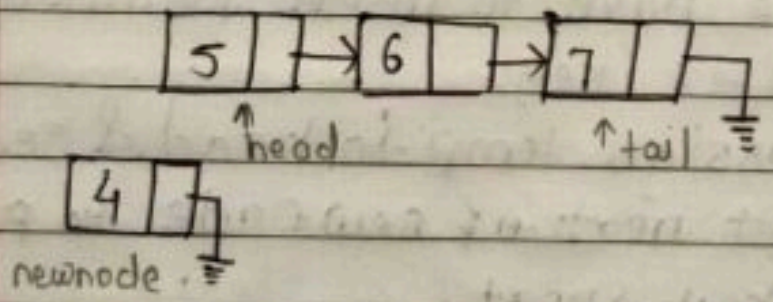
To create newnode

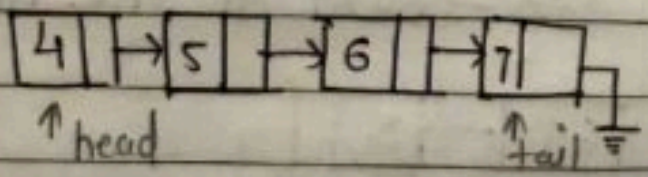
```
head -> next = nnode;
tail = nnode.
```

2] Insertion:-

a) Inserting at the front (beginning):

- To insert node at the beginning, we have to first create node.
- Then next pointer of newnode is pointed towards, the first (head) node of list
- The newnode is assigned as head node





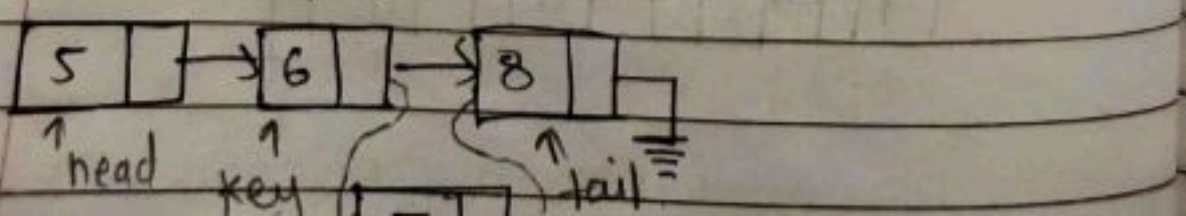
After inserting newnode at beginning

- pseudocode:

```
#create node

nnode = newnode
nnode → next = NULL
cout << "Enter data";
cin >> nnode → data;

#insert at beginning

nnode → next = head;
head = nnode.
```

b) Inserting after the node.

- Create a newnode
- Search node after which you want to insert new node. let we have to insert newnode after 'key'
- Assign temp to head & search key
- set next of new node to point to key → next.
- And key → next to the newnode.



pseudocode :-

create node.

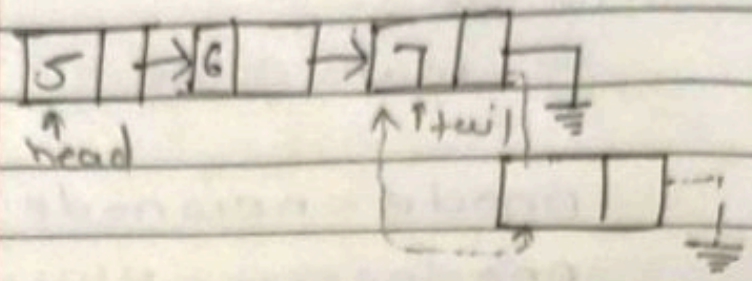
```
nnode = newnode;
nnode->next = NULL;
cout << "Enter data".
cin >> nnode->data;
```

insert after key

```
nnode *temp1, temp;
cout << "Enter node after which
you want insertion";
cin >> key;
temp = head
while (temp->data->data1 = key
        temp = temp->next,
temp1 = temp->next;
nnode->next = temp1;
temp->next = nnode.
```

c) Insertion at end:-

- Create a new node
- make linking between new node & tail node.
- Assign newnode as a tail node.



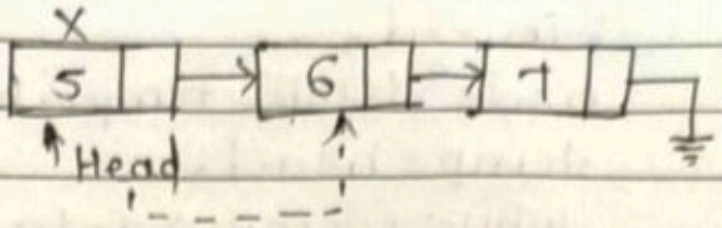
c) Inserting at last

pseudocode

```
#create newnode
nnode = new node;
nnode->next = NULL;
Cout << "Enter data";
cin >> nnode->data;

#Insert at end

nnode *temp;
temp = head;
while(temp->next != NULL)
        temp = temp->next;
temp->next = nnode;
```

3] Deletion:-

a) Deleting the first node
- Make second node as a head node
- Delete first node, using delet keyword.

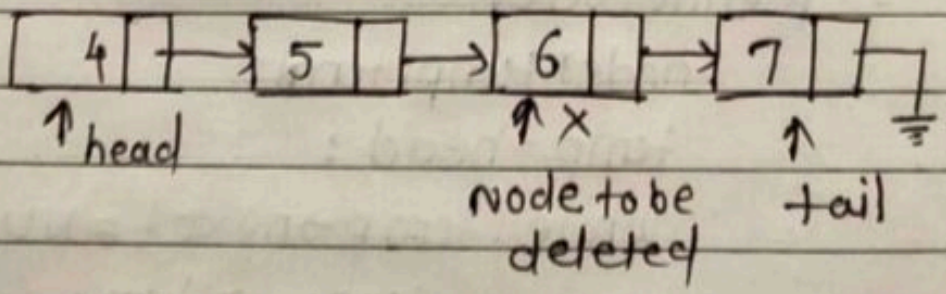


- pseudocode:

```
node *temp;
temp = head
head = head → next;
delet temp;
```

b) Deleting at middle / particular node.

- Here node to be deleted is searched
- Then we make linking between node previous & after to the node to be deleted.
- delete the node using delete keyword.



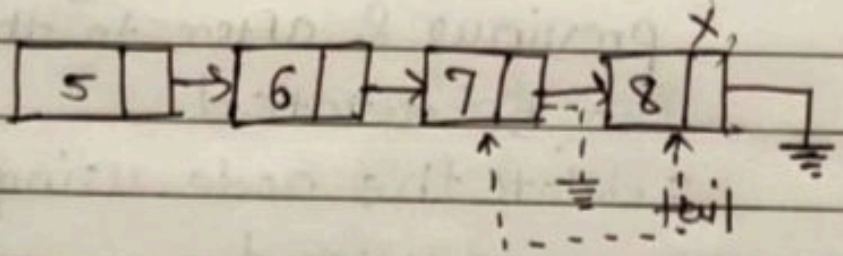
- pseudocode.

```
Cout << "After which node you want to delet";
```

```
cin>>d;
node* temp; temp1; temp2;
temp = head;
while (temp->data != d)
        temp = temp->next;
temp1 = temp->next;
temp2 = temp1->next;
temp->next = temp2;
delete temp1;
```

c) Deleting the last node:

- Here next of node before the last node is pointed towards NULL
- last node is deleted.

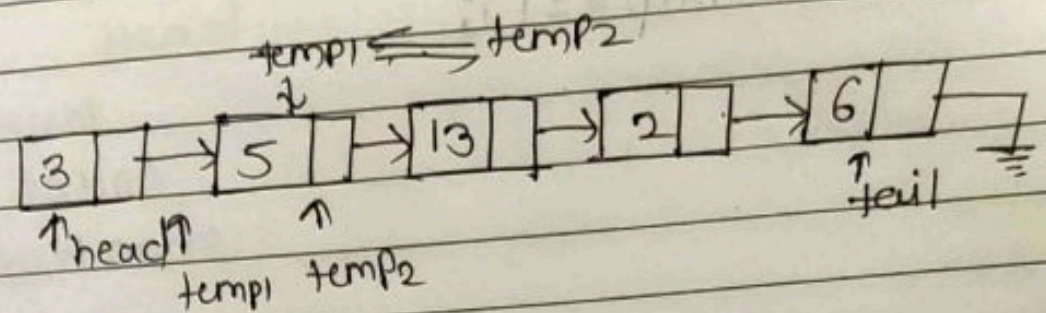


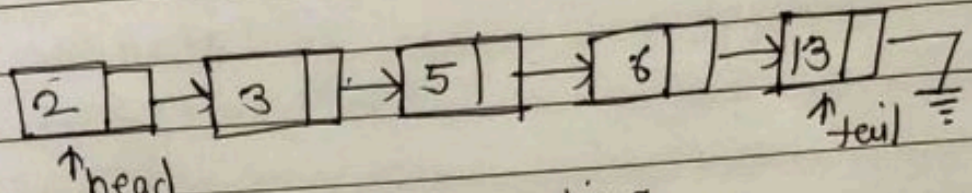
• pseudo code:

```
node* temp, temp1;
temp = head;
while (temp->next != NULL){
        temp1 = temp.
        temp = temp->next;
}
delet temp1.
```

5) Sorting:

- Here, we compare each node with its adjacent node. if next node is less than previous node, then values are swapped.
- for this, we need to traverse through entire linked list.





After sorting.

- pseudo code:

```
node * temp, * temp1, * temp2;

do
{
    swap = 0;
    temp1 = head;
    temp2 = temp1 -> next;
    while (temp1 -> datanext != NULL)
    {
        if (temp1 -> data > temp2 -> data)
        {
```