Date 8/2/19

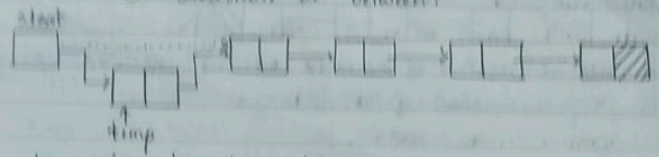
Insertion into a Linked list :-

It may be possible in two ways :

(i) Insertion at beginning.

(ii) Insertion in between.

i)



temp is ptr which points to the node which has to be inserted.

temp -> info = data ;

start points to the 1st element of linked list.

For insertion at beginning, we assign the value of start to the link part of inserted node

temp -> link = start;

Now, inserted node points to the next node which was beginning node of the linked list.

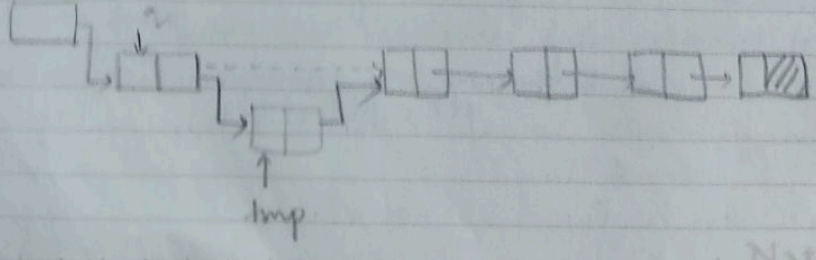
Now, the inserted node is the 1st node of the linked list. so, start will be assigned as -

start = temp;

Now, start will point to the inserted node which is the 1st node of the linked list. Dotted line represents the link before insertion.

i.e temp -> info = data ;
temp -> link = start ;
start = temp ;

ii)



Natural

11/4/14

1st we traverse the linked list for obtaining the node after which we want to insert the element. We obtain ptr 'q' which points to the element after which we have to insert new node. For inserting the element after the node. We give the link part of that node to the link part of inserted node. And the address of the inserted node is placed into the link part of the previous node.

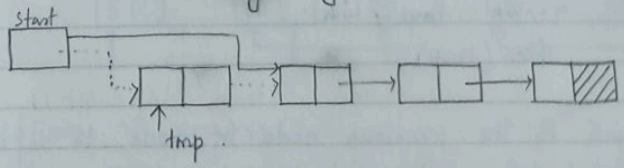
```
tmp -> info = data ;
tmp -> link = q -> link ;
q -> link   = tmp ;
```

Here, q is pointing the previous node. After statement 2, link of inserted node will point to the next node and after statement 3, link of previous node will point to the inserted node.

Deletion from a linked list :

For deleting a node from a linked list, 1st we traverse the linked list and compare with each element. After there may be 2 cases for deletion.

1) Deletion at beginning :



Here, start points to the 1st element of linked list. If element to be deleted is the 1st element of linked list then we assign the value of start to temp as

```
tmp = start ;
```

So, now tmp points to the 1st node which has to be deleted

Now, we assign the link part of the deleted node to start as,

```
start = start -> link;
```

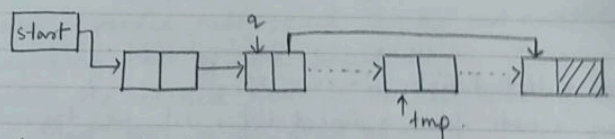
Since, start points to the 1st element of the linked list, so, start -> link will point to the 2nd element of linked list. Now, we should free the element to be deleted which is pointed by temp.

```
free (tmp);
```

so, the whole process for deletion of 1st element of linked list will be -

```
tmp = start;
start = start -> link;
free (tmp);
```

ii) Deletion in between :



If the element is other than the 1st element of the linked list, then we give the linked part of the deleted node to the linked part of the previous node. This can be as

```
tmp => q -> link;
q->link = temp -> link;
free (tmp);
```

Here, q points to the previous node of node to be deleted
After statement 1 temp will point to the node to be deleted.
After statement 2 link of previous node will point to the next node of the node to be deleted.
If the node to be deleted is the last node of the linked list then statement 2 will — q -> link = NULL;

Linked list.

```
struct node
        { int data;
          struct node * link;
        };
```

Here, member of the structure struct node * link points to the structure itself. This type of structure is called linked list.

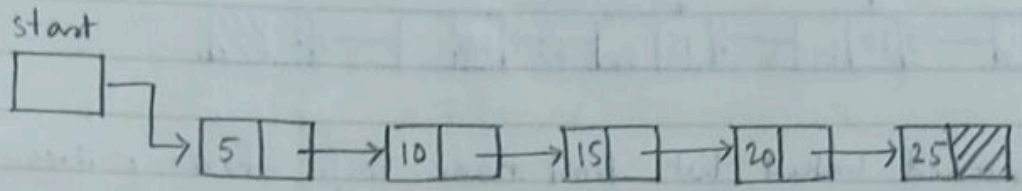
A linked list is a collection of elements called nodes. Each node contains two parts.

first part contains information field and

the second part contains the address of the next node.

The address part of last node of linked list will have NULL value.

Reversed Linked List :



Let us take a linked list. Now we want to reverse the linked list. Reverse of this linked list will do the following things :

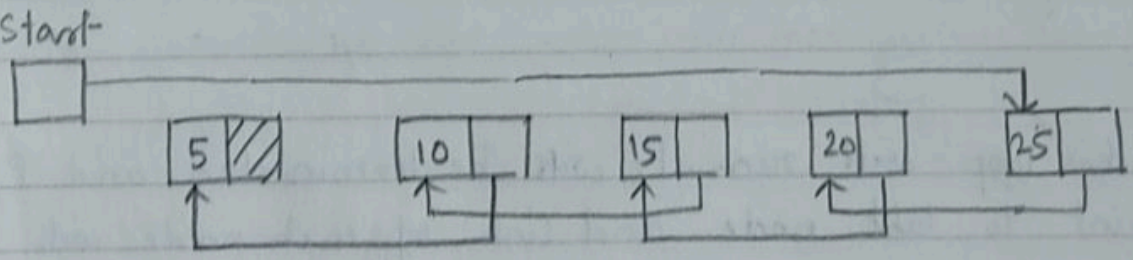
(i) First node will become the last node of linked list.

(ii) Last node will become the first node of linked list. And now, start will point to it.

(iii) Link of second node will point to first node, link of third node will point to second node and so on.

(iv) Link of last node will point to the previous node of last node in linked list.

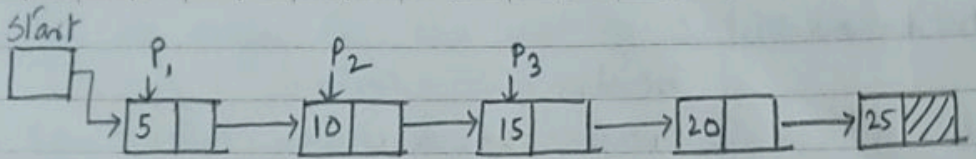
Now, the reversed linked list will be as,



Creation of reverse()

We will take 3 pointers P_1, P_2, P_3. Initially P_1, P_2 and P_3 will point to first, second and third node of linked list.

```
P1 = start;
P2 = P1 -> link;
P3 = P2 -> link;
```



Since in reversed linked list first node will become the last node, so the linked part of first node should be NULL.

$P_1 \rightarrow link = NULL;$

Now, link of second node should point to first node.

Hence, $P_2 \rightarrow link = P_1;$

Now, we will traverse the linked list with P_3 pointer and we shift pointers P_1 and P_2 forward. We assign P_1 to the linked part of P_2, so that link of each node will now point to its previous node.

```
while (P3 != NULL)
{
    P1 = P2;
    P2 = P3;
    P3 = P3 -> link;
    P2 -> link = P1;
}
```

When this loop will run P_3 will be terminated and P_2 will point to last node and link of each node will point to its previous node. Now, start will point to the last node of linked list which is first node of reverse linked list.

$start = P_2;$