Linked List

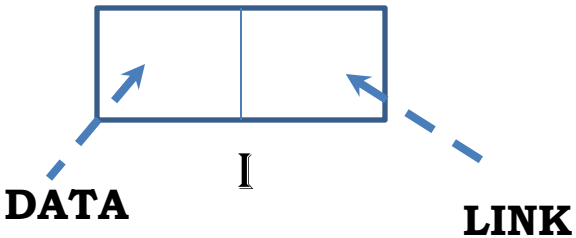
Singly linked list

Prof. Viswanathan V

VIT Chennai

Linked List

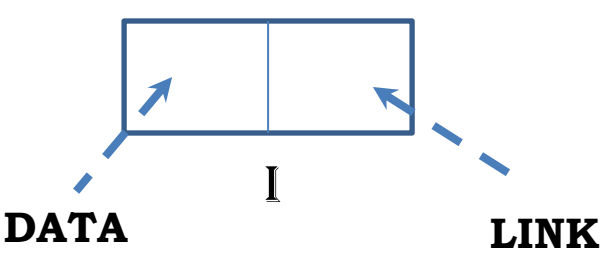
- Linked list a data structure that leads to algorithm that minimize data movement as insertions and deletions occurs in an ordered list.
- Each element called node. Each node has two fields called DATA and LINK

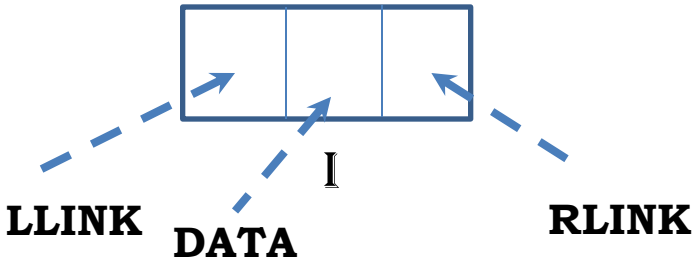


Linked List

Singly Linked List

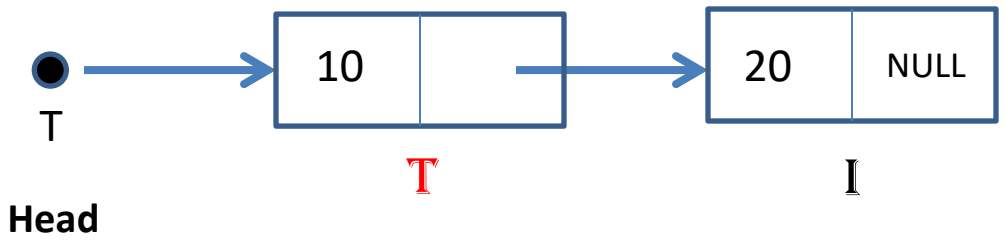
Doubly Linked List



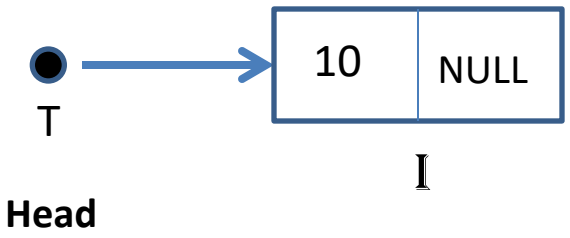


Creating linked list

1. *Algorithm Createlist(T)*
2. *{ // 'T' is the pointer to the first in the list*
3. *Get new node I*
4. *T = I*
5. *DATA(I) = 10*
6. *Get new node I*
7. *DATA(I) =20*
8. *LINK(T) = I*
9. *LINK(I) = NULL*
10. *}*



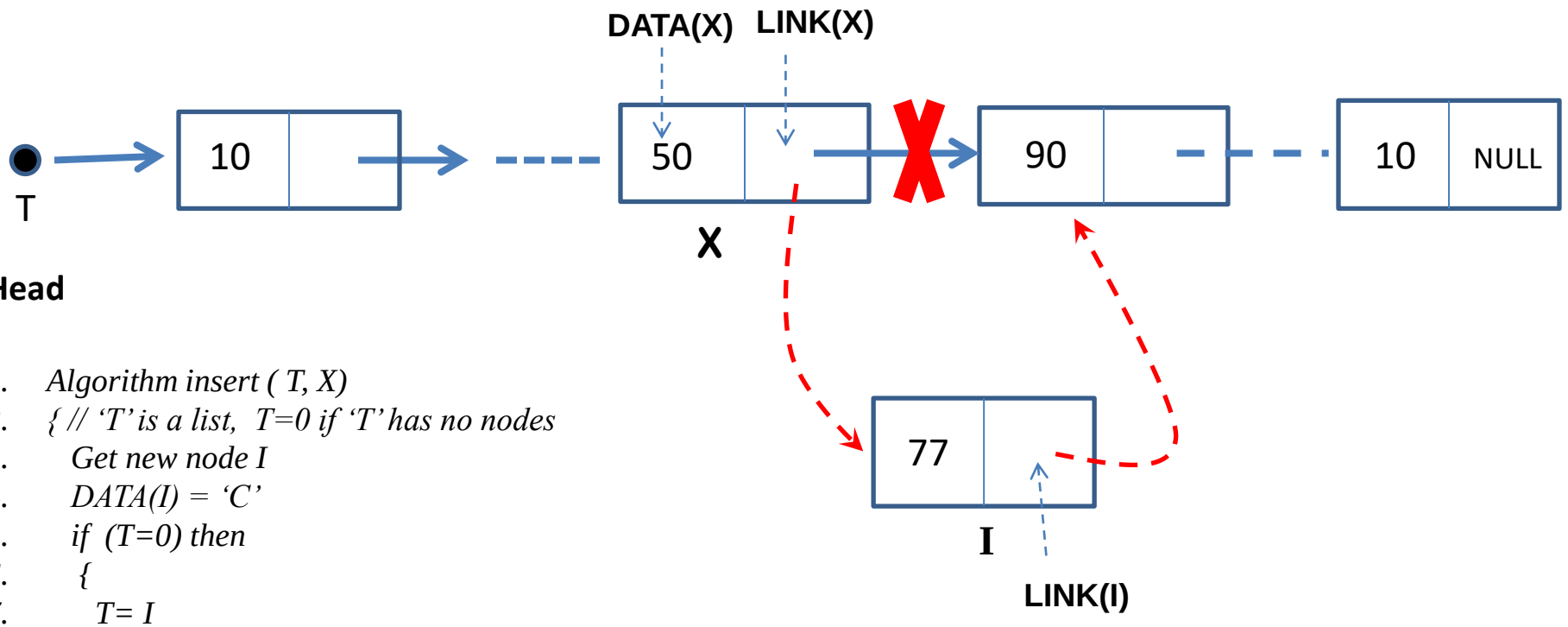
Inserting a node in a list - as first node in the list




```
1.  Algorithm insert ( T, X)
2.  { // 'T' is a list,  T=0 if 'T' has no nodes
3.     Get new node I
4.     DATA(I) = 'C'
5.     if  (T=0) then
6.      {
7.        T= I
8.        LINK(I) = NULL
9.      }
10.    else
11.    {
12.      LINK(I) = LINK(X)
13.      LINK(X) =  I
14.     }
15.  }
```

Inserting a node in a list –

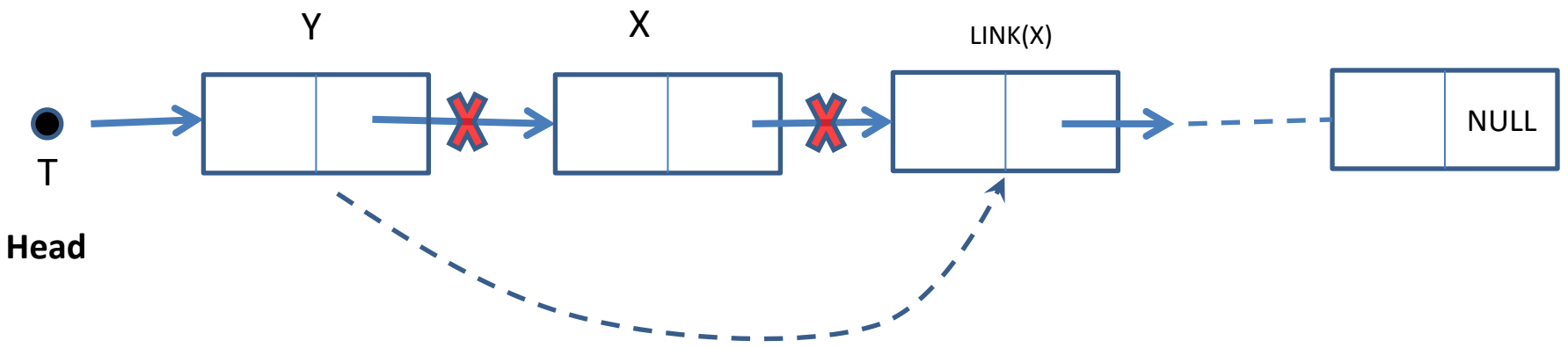
as intermediate node in the list after node X




```
1.  Algorithm insert ( T, X)
2.  { // 'T' is a list,  T=0 if 'T' has no nodes
3.     Get new node I
4.     DATA(I) = 'C'
5.     if  (T=0) then
6.      {
7.        T= I
8.        LINK(I) = NULL
9.      }
10.     else
11.     {
12.       LINK(I) = LINK(X)
13.       LINK(X) =  I
14.      }
15. }
```

Deleting a node in a list -

delete node 'X' from the list, 'Y' be the preceding node 'X'



1. *Algorithm Deletenode (X , Y, T)*
2. *{ // Y=0 if 'X' is the first of 'T'*
3. *if (Y=0) then T= LINK(X)*
4. *else*
5. *LINK(Y) = LINK(X)*
6. *Remove 'X' from storage*
7. *}*

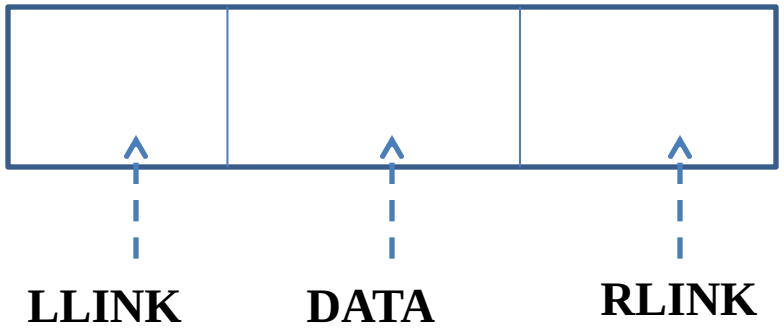
Code snippet in 'C'

```
//linked list node
struct Node
{
  int data;
  struct Node *next;
};
```

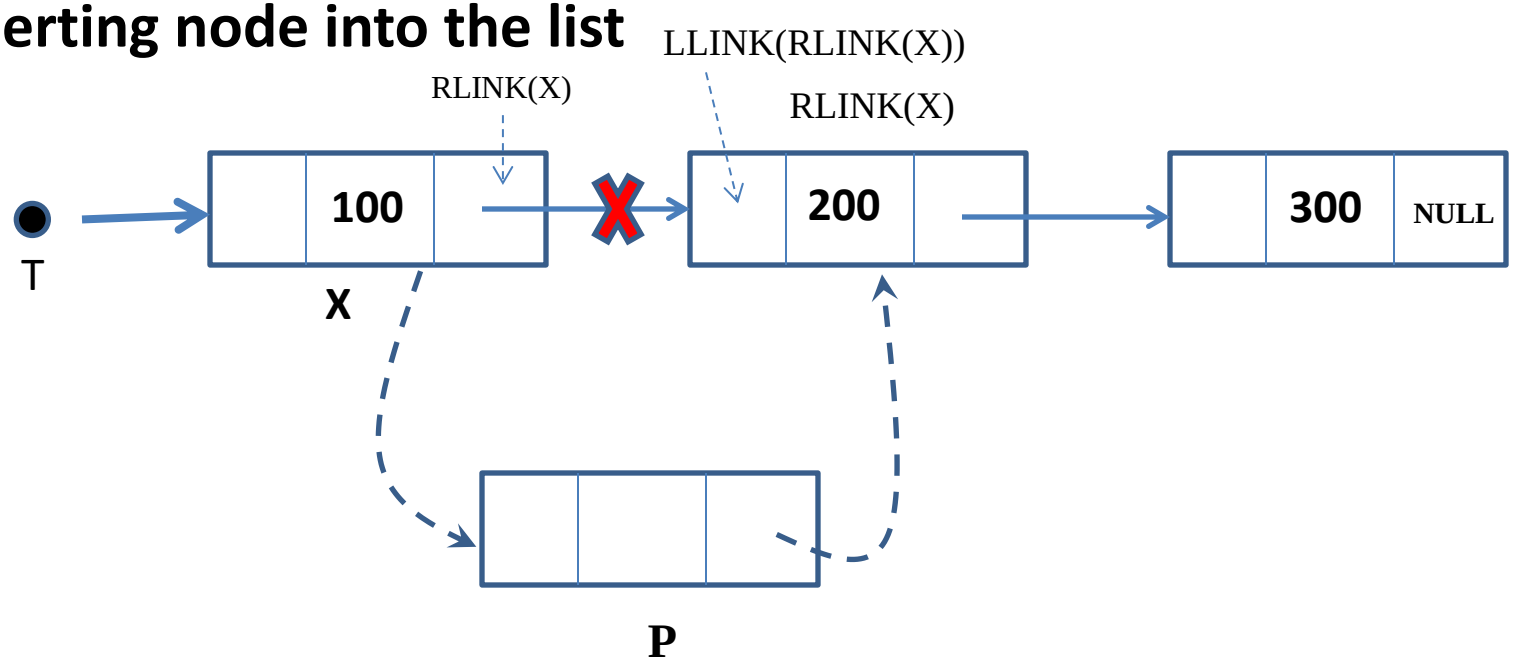
```
void insert(struct Node** head_ref, int new_data)
{  /*inserts a new node on the front of the list. */
    /* 1. allocate node */
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    /* 2. put in the data  */
    new_node->data  = new_data;
    /* 3. Make next of new node as head */
    new_node->next = (*head_ref);
    /* 4. move the head to point to the new node */
    (*head_ref)    = new_node;
}
```

Doubly linked list

A doubly linked list has two pointers, a forward link and a backward link. The forward link is points to the next node in the list, whereas the backward link points to the preceding node

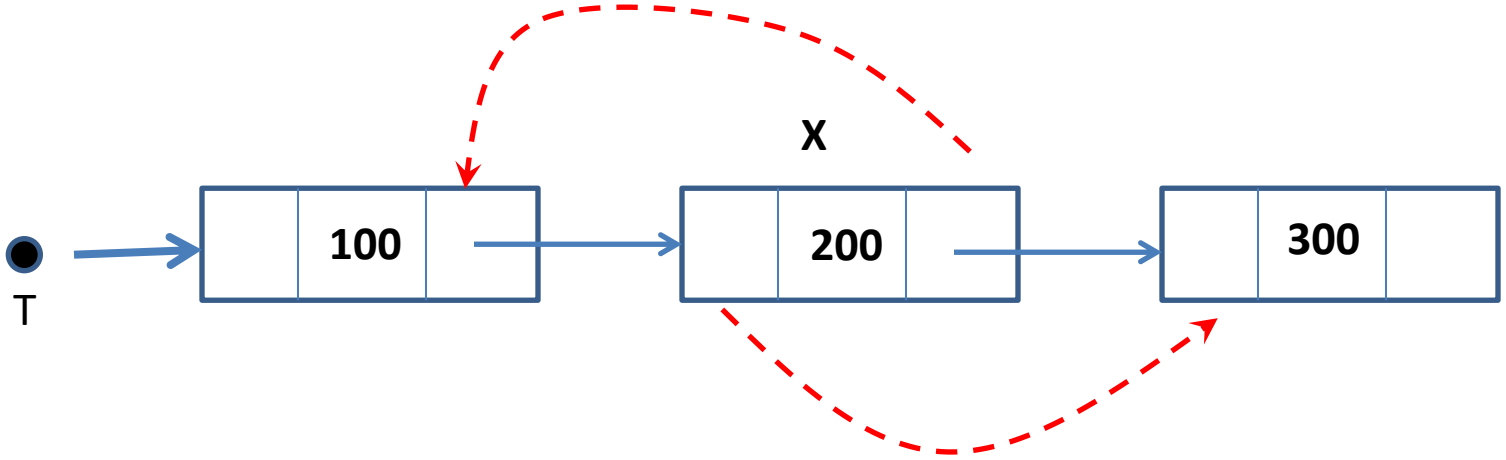


Inserting node into the list



1. *Algorithm Dinsert (P, X)*
2. *{ // insert node 'P' into right of node 'X'*
3. *LLINK(P) = X*
4. *RLINK(P) = RLINK(X)*
5. *LLINK(RLINK(X)) = P*
6. *RLINK(X) = P*
7. *}*

Deleting node from the list



1. *Algorithm Ddelete(X, T)*
2. *{*
3. *if (X=T) then print "no-more- node" ; return*
4. *RLINK(LLINK(X)) = RLINK(X)*
5. *LLINK(RLINK(X)) = LLINK(X)*
6. *Remove X from storage*
7. *}*


Thank
you