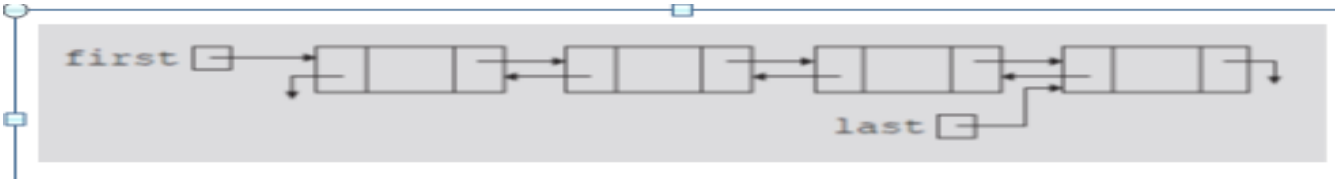
In a doubly linked list:

- Every node has two links: one points to the next node and one points to the pervious node.
- Insertions and deletions now require more operations but other operations are simplified.
- The list can be traversed in either direction.
- Two “start pointers”- first element and last element are available.



Inserting a node at the end of a doubly linked list

1) A new node is created, and then its three fields are initialized
2) The info field is populated
3) The next field is null.
4) The prev field points to the tail
5) The next field of tail (its predecessor) is set to reference the new node
6) Tail is set to reference the new node.

Example 1 (Creating a simple doubly linked list)

```
struct node {
     int a;
     node *next;
     node *prev;
};

int main(){
    struct node  *head, *tail, *current;
    struct node *first, *second, *third,;

    //create some nodes
    first=new node;
    second=new node;
    third=new node;

    //populate with data
    first->data=1;
    second->data=2;
    third->data=3;

    //point them at each other
    first->prev=NULL;
    first->next=second;
    second->prev=first;
    second->next=third;
    third->prev=second;
```

```
    third->next=NULL;

    //set the head and tail
    head=first;
    tail=third;

    //traverse from beginning to end
    current = head;
    while (current != NULL) {
       cout<< current->data;
       current = current->next;
    }

    //traverse from end to beginning
    current = tail;
    while (current != NULL) {
       cout<< current->data;
       current = current->prev;
    }
}
```

Challenge:

Change the syntax such that you have head, tail and current as your only variables.

Example 2 (Using a general algorithm to insert)

```
struct node {
     int a;
     node *next;
     node *prev;
};

void insertBeg (int item,struct node *&head,struct node *&tail){
    node * newNode=new node;
    newNode->a=item;
    newNode->prev=NULL;//Since there is nothing behind it has to be null

    if (!head){
        newNode->next=NULL;

       //This is the first one to be inserted
       //tail will also point to this one.
       tail=newNode;
    }else{
       //The new node is always being placed in the beginning
       newNode->next=head;

       //The old node needs to be placed ahead.
       //So the previous of the old node which is the head->prev needs
       //to point to the new node
       head->prev=newNode;
```

```
    }
    //The head is now pointing to the new node
    head=newNode;
}

int main(){
    struct node  *head=NULL, *tail=NULL;
    insertBeg(5,head,tail);
    insertBeg(15,head,tail);
}
```

Questions to consider?

1) How do you insert at the end?

2) How do you insert in the middle?

3) What happens when you delete a node from the middle?

4) What happens when you delete a node from the end?

5) What happens when you delete a node from the beginning?

6) What happens when you delete the only node?