**Program: Find Middle Element in Linked List using only one travels**

**( without count)**

void LinkedList::findMiddle()

{

Node \*slow=head;

Node \*fast=head;

while(fast->next!=NULL && fast->next->next!=NULL)

{

fast=fast->next->next;

slow=slow->next;

}

cout<<"Middle Element is "<<slow->data<<endl;

}

**Note: This Logic can be used to Detect the loop in the list**

**Program: Reverse Linked List**

void LinkedList::reverseList()

{

Node \*first=head;

head=head->next;

Node \*second=head;

**first->next=NULL;**

while(head!=NULL)

{

head=head->next;

second->next=first;

first=second;

second=head;

}

**head=first;// bz head contain NULL after termination of loop**

}

**Program: Sort Linked List**

**We use selection sort in the following way**

void LinkedList::sort()

{

Node \*trav=head;

while(trav!=NULL)

{

Node \*first=trav;

Node \*second=trav->next;

while(second!=NULL)

{

if(first->data>second->data)

{

int temp=first->data;

first->data=second->data;

second->data=temp;

}

second=second->next;

}

trav=trav->next;

}

}

**Program: Detect the loop in the list**

**Floyd’s Cycle-Finding Algorithm:** This is the fastest method and has been described below:

* Traverse linked list using two pointers.
* Move one pointer(slow\_p) by one and another pointer(fast\_p) by two.
* If these pointers meet at the same node then there is a loop. If pointers do not meet then linked list doesn’t have a loop