May -29 LL

29 May 2020 21:17

Task:

1. Complete old tasks
2. Optimize insertAtPosition function to handle position greater than length+1.
3. Delete a node with the given data.
4. Reverse a LL
5. Push to Github

Insertion at nth position
Delete a LL
Delete node from beginning of a LL
Delete node from end of a LL
Delete node from nth position
Find the middle element of a LL
Reverse a Linked List

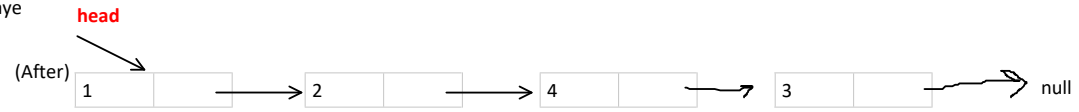
3. Insertion at nth position

insertAtPosition(head,data,pos)
insertAtPosition(head,4,3)

1st Position - New Node must be the head
2nd Position - 2nd node ban jaye

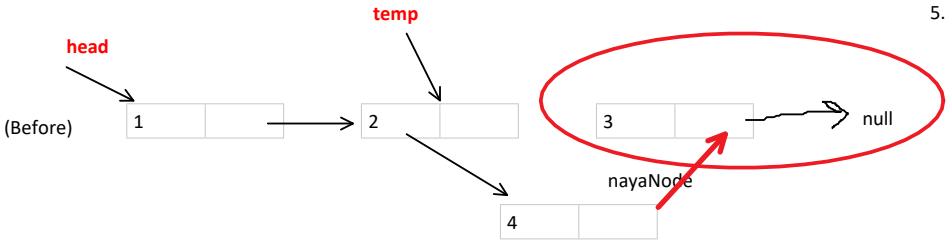
3->1 jump

Nth position
n-2 jumps



Intermediate:

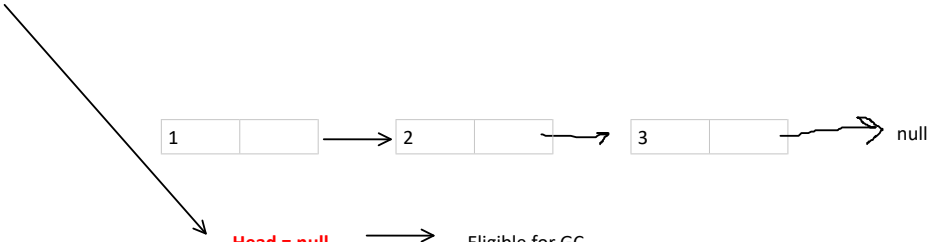
1. Add new Node
2. Create temp=head;
3. Take n-2 jumps
4. nayaNode.next=temp.next;
5. Temp.next=nayaNode;

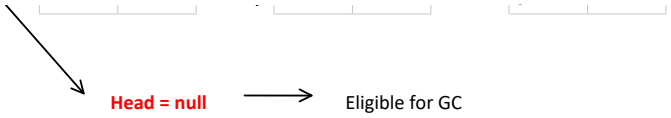


(If Position = 1) - We will have to handle separately

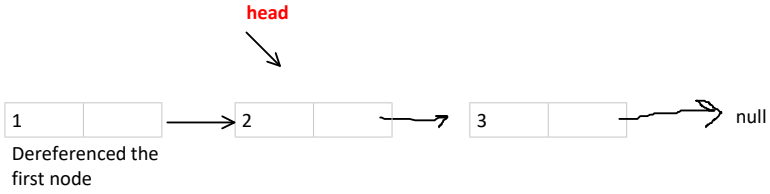
Nya node
Naya node.next = head;
Head ko naye node se change(replace)

Deletion:
Delete the LL





Delete a node at the beginning: [head=head.next;]



Find the Middle Element (Print the second in case of even elements)

1->2->3->4->5->null // 3

1->2->3->4->5->6->null // 4

5/2 = 2
6/2 = 3

Floor Value
Ceil Value

Value: 1.6

Floor: 1
Ceil: 2

Value: 5.2
Floor: 5
Ceil: 6

Method:
Find Length.
Traverse till n/2 and print value;

Find the Middle Element (Print the second in case of even elements) - (Method - 2)

20 km/hr

A B

10 km/hr

100 kms

When rabbit reaches end tortoise will be at middle position, because the speed of rabbit is double that of tortoise.

fptr

1->2->3->4->5->null - Odd Number of elements

At each step:
Fptr takes 2 jumps
Sptr takes 1 jump

```
while(fptr.next!=null)
{
        fptr=fptr.next.next;
        sptr=sptr.next;
}
```

fptr

1->2->3->4->5->6->null - Even Number of elements

sptr

```
while(fptr!=null)
{
        fptr=fptr.next.next;
        sptr=sptr.next;
}
```

```
while(fptr.next!=null && fptr!=null)
{
        fptr=fptr.next.next;
        sptr=sptr.next;
}
Syso(sptr.data);
```

Reverse the Linked List (Using Loops)

1->2->3->4->5->null

Next

prev
next

Head,

prev

(Before)

1 2 3 current 5

prev=null

Next=2

->null

(After)

Null<-

1 2 3 4 5

head

Next = null

Prev = 2

Current = null

prev

```
Node current = head;
Node prev=null;
Node next=null;
while(current!=null)
{
        next=current.next; // 2
        //swapping
```

```
        Current.next=prev;
        //now we have to move current ahead
        prev=current;
        current=next;
}

//after the loop, current will be null and prev will be at the last position
which is the head for the new LL

Return prev;
```