Question : Make a Linked list & add the following elements to it:(1,5,7,3,8,2,3).Search for the number 7 & display the index.

```
class ll
{
      Node head;
    public void add(int d)
    {
        Node newNode=new Node(d);
        if(head==null)
        head=newNode;
        else
        {
            Node lastNode=head;
            while(lastNode.next!=null)
            lastNode=lastNode.next;
            lastNode.next=newNode;
        }
    }
    public void display()
    {
       Node currentNode = head;
         while (currentNode != null)
             {
               System.out.print(currentNode.data + "\t");
               currentNode = currentNode.next;
             }
    }
    public void search(int d)
    {
       int k=0;
       Node currentNode=head;
```

```
        if(head!=null)
        {
           while(currentNode.data!=d)
           {
           currentNode=currentNode.next;
           k++;
           }
        }
        if(head==null)
        System.out.println("list is empty");
        else
        System.out.println(k+1);
    }
}
class  Node
{
    int data;
    Node next;
    Node(int d)
    {
        data=d;
        next=null;
    }
}
class oop
{
    public static void main(String[] args)
    {
        ll ll=new ll();
        ll.add(1);
        ll.add(5);
```

```
        ll.add(7);
        ll.add(3);
        ll.add(8);
        ll.add(2);
        ll.add(3);
        ll.display();
        System.out.print("\n");
        ll.search(7);
    }
}
```

Output:

1 5 7 3 8 2 3

3

Question :Take a elements(numbers in the range of 1-50) of a Linked list as input from the user . Delete all nodes which have values greater than 25.

```
class ll
{
      Node head;
    public void add(int d)
    {
        Node newNode=new Node(d);
        if(head==null)
        head=newNode;
        else
        {
            Node lastNode=head;
            while(lastNode.next!=null)
            lastNode=lastNode.next;
            lastNode.next=newNode;
```

```
        }
    }
    public void display()
    {
        Node currentNode = head;
          while (currentNode != null)
             {
               System.out.print(currentNode.data + "\t");
               currentNode = currentNode.next;
             }
    }
   public void func()
   {
      while(head.data>25)
      head=head.next;
      //Node pvNode=head;
      Node currentNode=head;
      while(currentNode.next!=null)
      {
         if(currentNode.next.data>25 )
         {
         currentNode.next=currentNode.next.next;
         }
         else
         currentNode=currentNode.next;
      }
   }
}
class  Node
{
    int data;
```

```
    Node next;
    Node(int d)
    {
        data=d;
        next=null;
    }
}
class Main
{
    public static void main(String[] args)
    {
        ll ll=new ll();
        ll.add(56);
        ll.add(15);
        ll.add(51);
        ll.add(17);
        ll.add(90);
        ll.add(67);
        ll.add(89);
        ll.add(19);
        ll.add(14);
        ll.add(45);
        ll.add(95);
        ll.add(100);
        ll.display();
        System.out.print("\n");
        ll.func();
        ll.display();
    }
}
```

Output:56 15 51 17 90 67 89 19 14 45 95 100

15 17 19 14

Question-Make insertion and deletion function for linked list in c++.

```
#include<iostream>
using namespace std;
class Node
{
	public :
		int data;
		Node *next;
	//constructer
	Node(int data)
	{
		this->data = data;
		this->next = NULL;
	}
	~Node()
	{
		int value = this->data;
		if(this->next != NULL)
		{
			delete next;
			this->next = NULL;
		}
		cout<<"Memory is free"<<endl;
	}
};
//Node insert at head
void InsertAtHead(Node* &head, int d)
```

```
{
	Node* temp = new Node(d);
	temp->next = head;
	head = temp;
}
void InsertAtTail(Node* &tail, int d)
{
	Node* temp = new Node(d);
	tail->next = temp;
	tail = tail->next;
}
void InsertAtPosition(Node* &tail, Node* & head,int position, int d)
{
	//insert at start
	if(position ==1)
	{
		InsertAtHead(head,d);
		return;
	}
	Node* temp = head;
	int c=1;
	while(c<position-1)
	{
		temp = temp->next;
		c++;
	}
	//end tail
	if(temp->next == NULL)
	{
		InsertAtTail(tail,d);
		return;
```

```
        }
        Node*  nodetoinsert = new Node(d);
        nodetoinsert->next = temp->next;
        temp->next = nodetoinsert;
}
//print list
void print(Node* &head)
{
        Node* temp = head;
        while(temp != NULL)
        {
                cout<<temp->data<< " ";
                temp = temp->next;
        }
        cout<<endl;
}
void deleteNode(int position, Node* &head)
{
        if(position == 1)
        {
                Node* temp = head;
                head = head->next;
                temp->next = NULL;
                delete temp;
        }
        else
        {
                Node* curr = head;
                Node* prev = NULL;
                int c=1;
                while(c< position)
```

```
                {
                        prev = curr;
                        curr = curr->next;
                        c++;
                }
                prev->next = curr->next;
                curr->next = NULL;
                delete curr;
        }
}
int main()
{
        Node* node1 = new Node(98);
        cout<< node1->data <<endl;
        // head pointedd to node1
        Node* head = node1;
        Node* tail = node1;
        InsertAtHead(head,76);
        print(head);
        InsertAtHead(head,12);
        print(head);
        InsertAtTail(tail,28);
        print(head);
        InsertAtTail(tail,6);
        print(head);
InsertAtPosition(tail,head,4,167);
        print(head);
        cout<<"Head "<<head->data <<endl;
        cout<<"Tail "<<tail->data <<endl;
        deleteNode(1,head);
        print(head);
```

```
        deleteNode(3,head);
        print(head);
        cout<<"Head "<<head->data <<endl;
        cout<<"Tail "<<tail->data <<endl;
        return 0;
}
```

Output:

```
98
76  98
12  76   98
12   76   98   28
12   76   98   28   6
12   76   98   167  28 6
Head 12
Tail 6
Memory is free
76 98 167 28 6
Memory is free
76  98  28  6
Head 76
Tail 6
```

Question-Make insertion and deletion function for linked list in java.

```
class Node
{
   int data;
   Node next;
   Node(int d)
   {
      data=d;
```

```
        next=null;
    }
}
class oop
{
    public static void main(String[] args)
    {
        ll ll=new ll();
        ll.add(23);
        ll.add(9);
        ll.add(23);
        ll.add(17);
        ll.add(98);
        ll.add(23);
        ll.display();
        System.out.print("\n");
        ll.delete(98);
        ll.display();
         System.out.print("\n");
       ll.insert(3,1);
       ll.insert(5,3);
       ll.insert(99,1);
       ll.insert(5,8);
       ll.insert(76,10);
       ll.display();
         System.out.print("\n");
    }
}
class ll
{
    int s=0;
```

```
 Node head;
 public void display()
 {
    Node currentNode=head;
    while(currentNode!=null)
    {
       System.out.print(currentNode.data+"\t");
       currentNode=currentNode.next;
    }
 }
 public void add(int d)
 {

  Node newNode=new Node(d);
  if(head==null)
  head=newNode;
  else
  {
     Node lastNode=head;
     while(lastNode.next!=null)
     lastNode=lastNode.next;
     lastNode.next=newNode;
  }
  s++;
 }
public void insert(int d,int k)
 {
    Node newNode=new Node(d);
    if(head==null)
      {
         head=newNode;
```

```
        s++;
        return;
      }
    if(k==1)
    {
      newNode.next=head;
      head=newNode;
    }
    else if(k==s+1)
    {
                Node currentNode=head;
                while(currentNode.next!=null)
                        currentNode=currentNode.next;
                currentNode.next=newNode;
    }
    else
    {
       Node currentNode=head;
    while(k-->2)
    {
       currentNode=currentNode.next;
    }
    //if(currentNode.next.next!=null)
    newNode.next=currentNode.next;
     currentNode.next=newNode;
    }
     s++;
}
public void delete(int d)
{
    Node currentNode=head;
```

```
        while(currentNode.next!=null)
        {
           if(currentNode.next.data!=d)
           currentNode=currentNode.next;
           else
           currentNode.next=currentNode.next.next;
        }
        if(currentNode.data==d)
        currentNode=null;
        currentNode=head;
        if(currentNode.data==d)
        head=head.next;
        s--;
     }
}
```

Output:

```
23  9  23  17  98  23
23  9  23  17  23
99  3  23  5  9  23  17  5  23  76
```