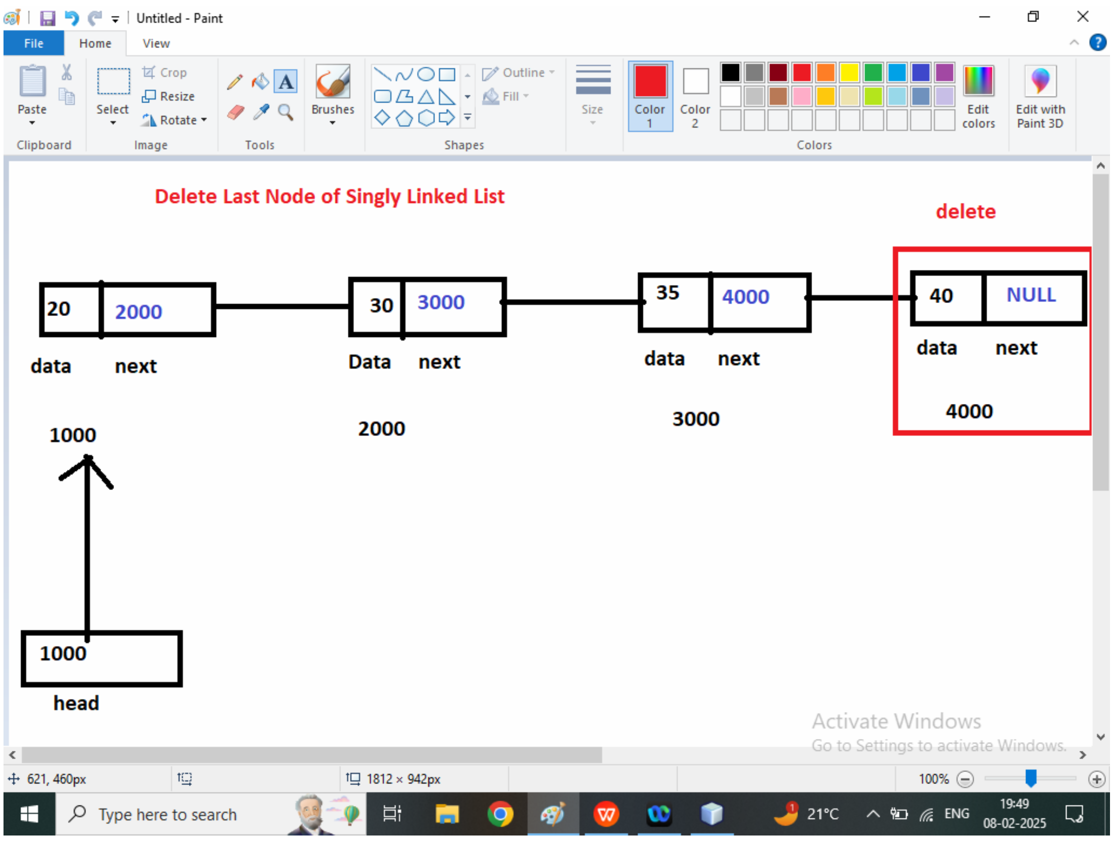
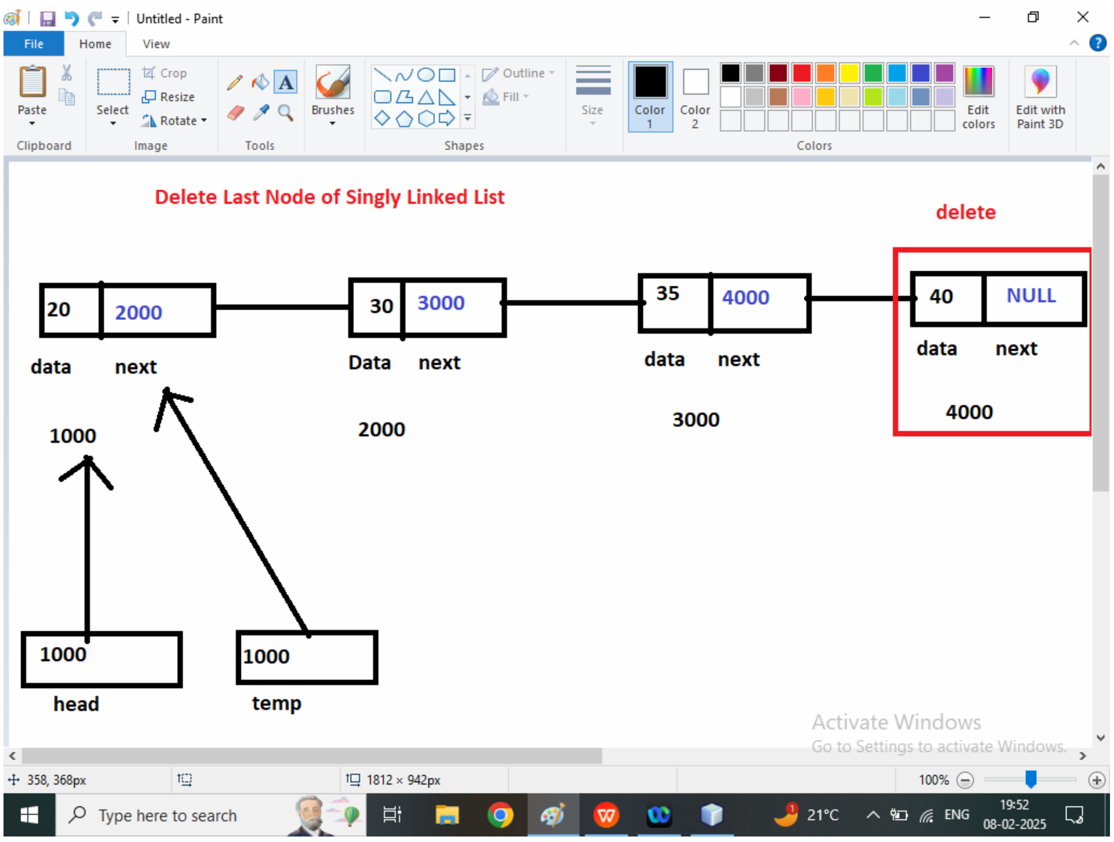
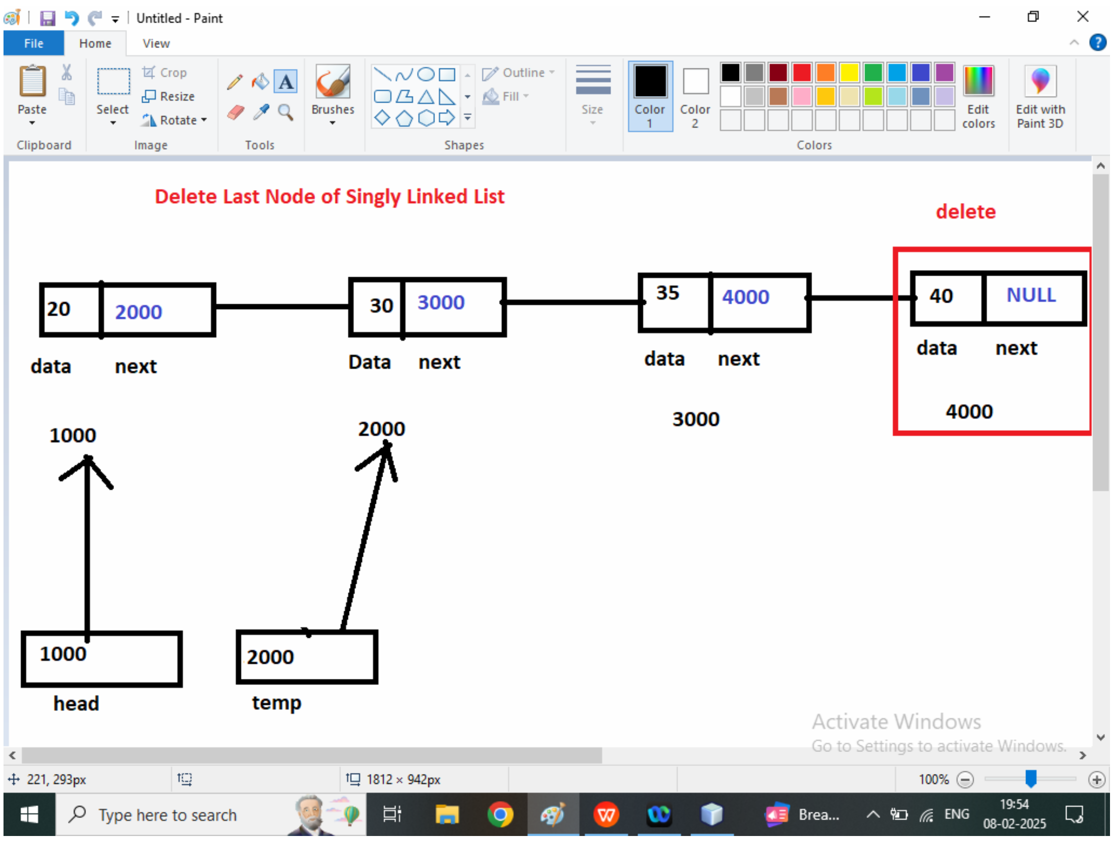
Q1.Write a java program to delete last node of singly linked List?

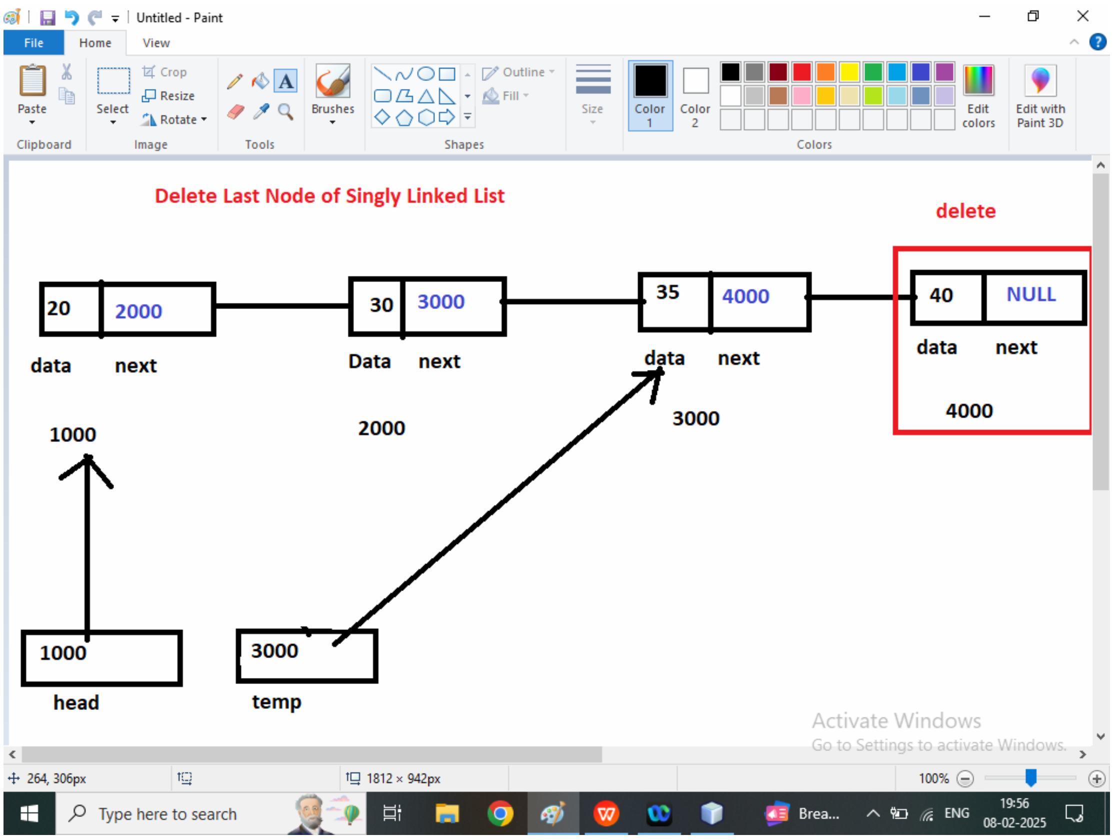


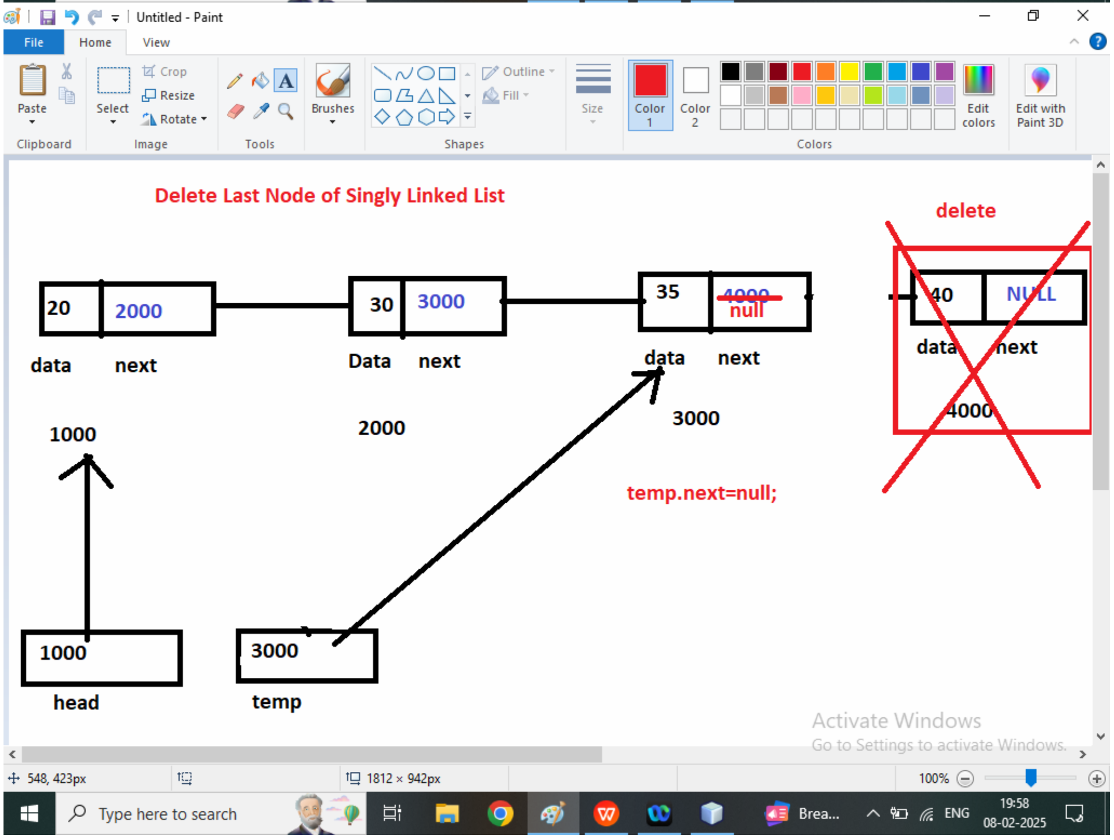


First Iteration



After Second Iteration





```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package dsafeb2025;

/**
 *
 * @author Admin
 */
public class Node {

    int data;
    Node next;

    public Node(int data) {
        this.data = data;
        next = null;
        System.out.println("Node created Success");
    }
public void display(Node head){
    Node temp=head;
    while(temp!=null){
        System.out.print("---->"+temp.data);//10--->20--->30--->40
        temp=temp.next;
    }
}

public Node addNodeAtStart(Node head,int data){
  //step1: Create a new Node
  Node newNode=new Node(data);
  //step2: Make Point NewNode to Current Node
  newNode.next=head;
  //step3: Update head
  head=newNode;
  //step4: return new head

    return head;

}
public void addNewNodeAtEnd(Node head,int data){
    //step1:Create new Node
    Node newNode=new Node(data);
    //step2: Traverse the list
    Node temp=head;
    while(temp.next!=null){
        temp=temp.next;
    }
    //step3: Setting the last node next pointer to the new node
    temp.next=newNode;


}
public void addNewNodeAtPos(Node head,int data,int pos){
    //step1: Create a  new Node
    Node newNode=new Node(data);
    //step2: Traverse the list upto specified position
    pos--;//3
    Node temp1=head;
    Node temp2=head.next;
    while(pos>1){
        temp1=temp1.next;
        temp2=temp2.next;
        pos--;//1
    }
    temp1.next=newNode;
    newNode.next=temp2;

}
public boolean isEmpty(Node head){
    return head==null;
}

public void deleteLastNode(Node head){
    //step1: check list is empty
    if(isEmpty(head)){
        System.out.println("List is Empty");
    }else{
        //step2: Traverse The List Upto Last Node
        Node temp=head;
        while(temp.next.next!=null){
            temp=temp.next;
        }
        //step3: To Update temp.next=null
        temp.next=null;
    }
}
public Node deleteFirstNode(Node head){
    //step1: Check the list is empty
    if(isEmpty(head)){
        System.out.println("List is Empty");
    }else{
        //step2: Store head into temporary variable
        Node temp=head;
        //step3: Move head to the next node
        head=head.next;
        //step4: Free memory of the temporary variable
        temp=null;

    }
    return head;
}

    public static void main(String[] args) {
        Node first = new Node(10);
        Node second = new Node(20);
        Node third = new Node(30);

        //Head point the first node of singly Linked List
        Node head=first;
        first.next=second;
        second.next=third;

        System.out.println("Print Data of Singly Linked List");
        System.out.println("===>"+first.data+"===>"+second.data+"===>"+third.data);
        System.out.println("Print Data of Singly Linked List Using head");
        System.out.print("===>"+head.data);
        System.out.print("===>"+head.next.data);
        System.out.print("===>"+head.next.next.data);

        System.out.println("Print Data of Singly Linked List Using Method");
        head.display(head);
       // head=head.addNodeAtStart(head, 5);
        System.out.println("\nPrint Data after New Node at starting in singly linked List");
        head.display(head);
        System.out.println("print Data After Add new Node at end of singly linked List");
        head.addNewNodeAtEnd(head, 40);
        head.display(head);
        System.out.println("\nPrint Data after insert new Node at Specific position\n");
        head.addNewNodeAtPos(head,35,4);
        head.display(head);
        head=head.deleteFirstNode(head);
        System.out.println("\nPrint data of singly Linked List After deletion first Node \n");
        head.display(head);
        head.deleteLastNode(head);
        System.out.println("\nPrint data of singly Linked List After deletion Last Node \n");
        head.display(head);

    }
}
```

Q3.Write a java program to delete specific node in singly Linked List?

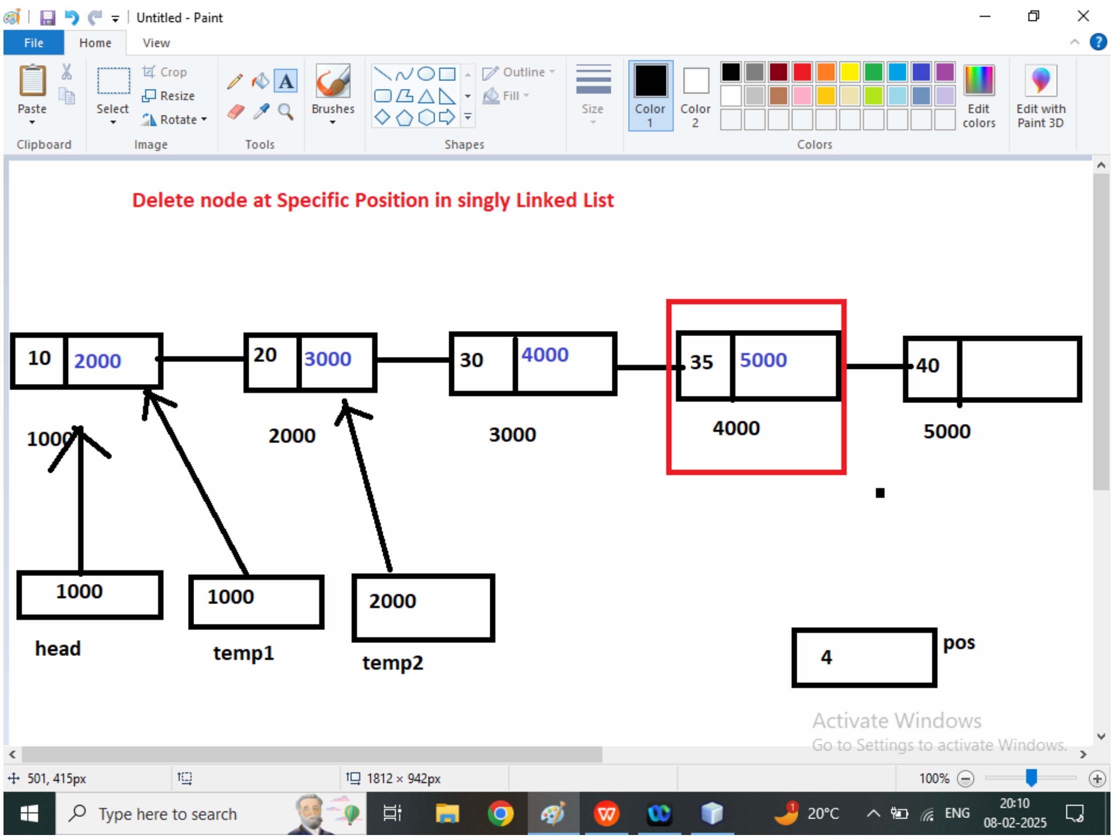
Algorithm:

Step1: Check List empty or not

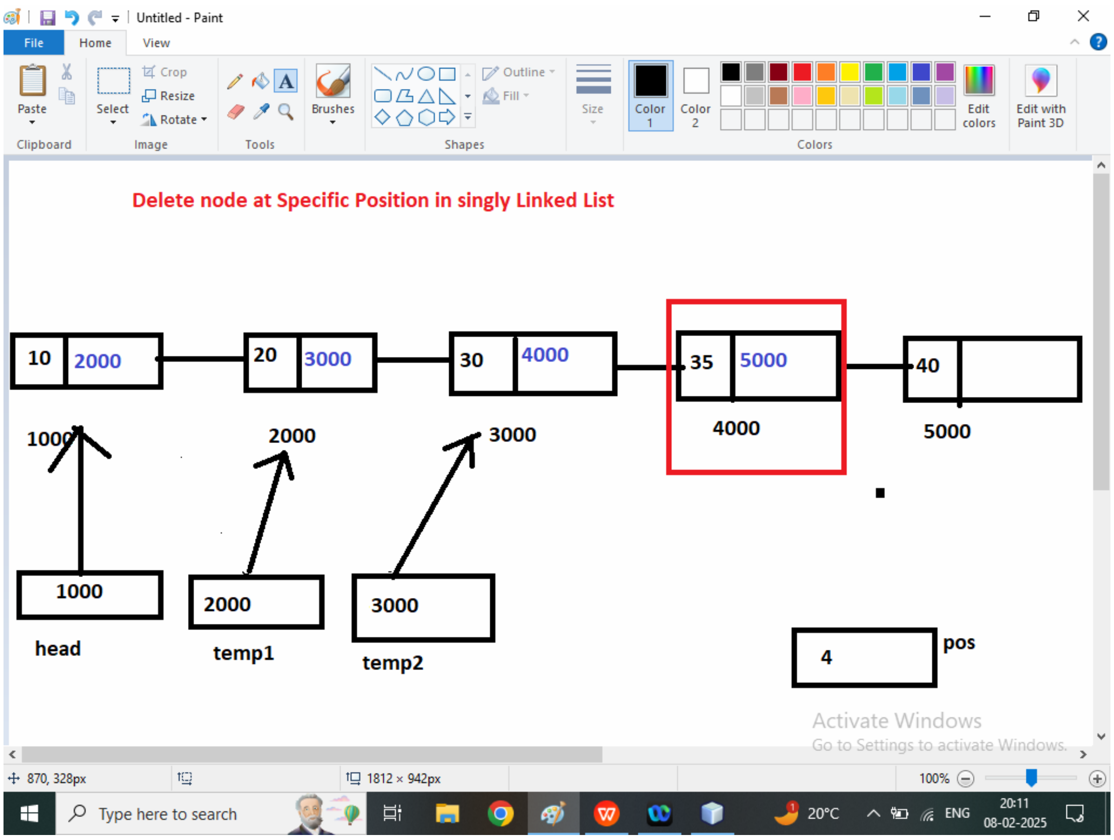
Step2: Traverse the List upto given position

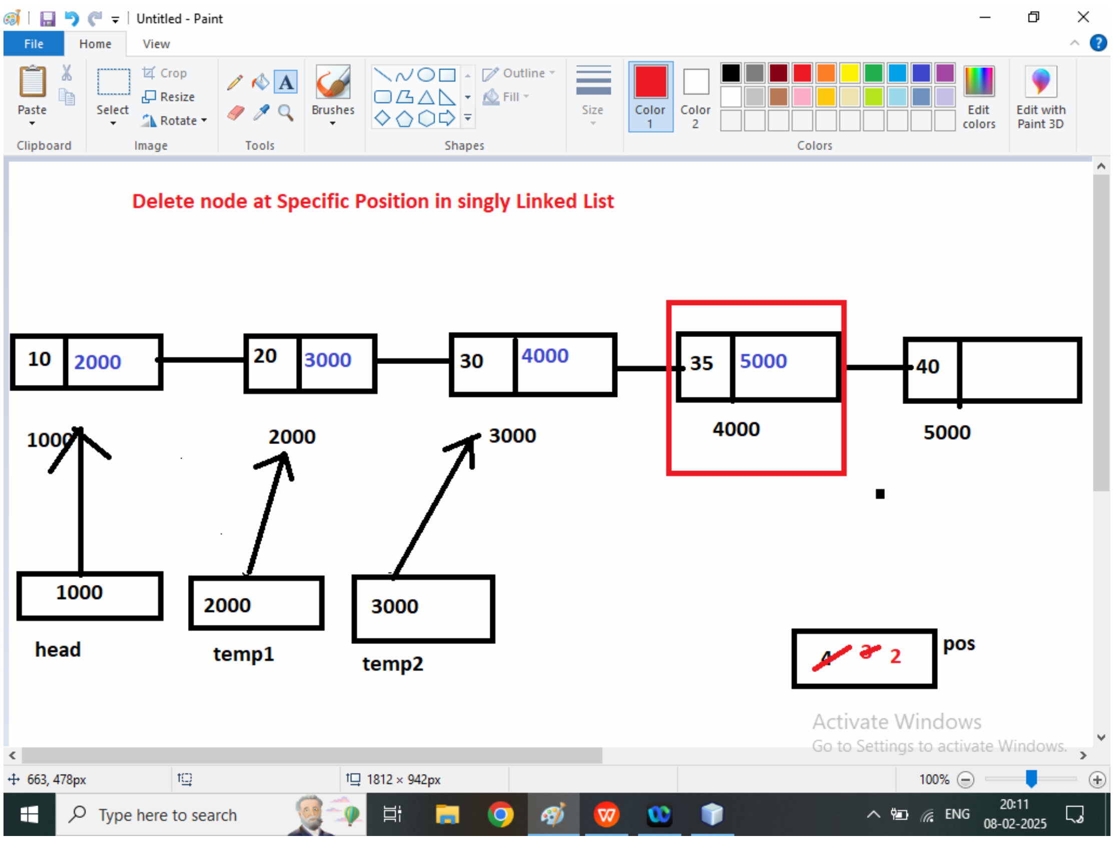
Step3: connect temp1.next=temp

Before Fist Iteration

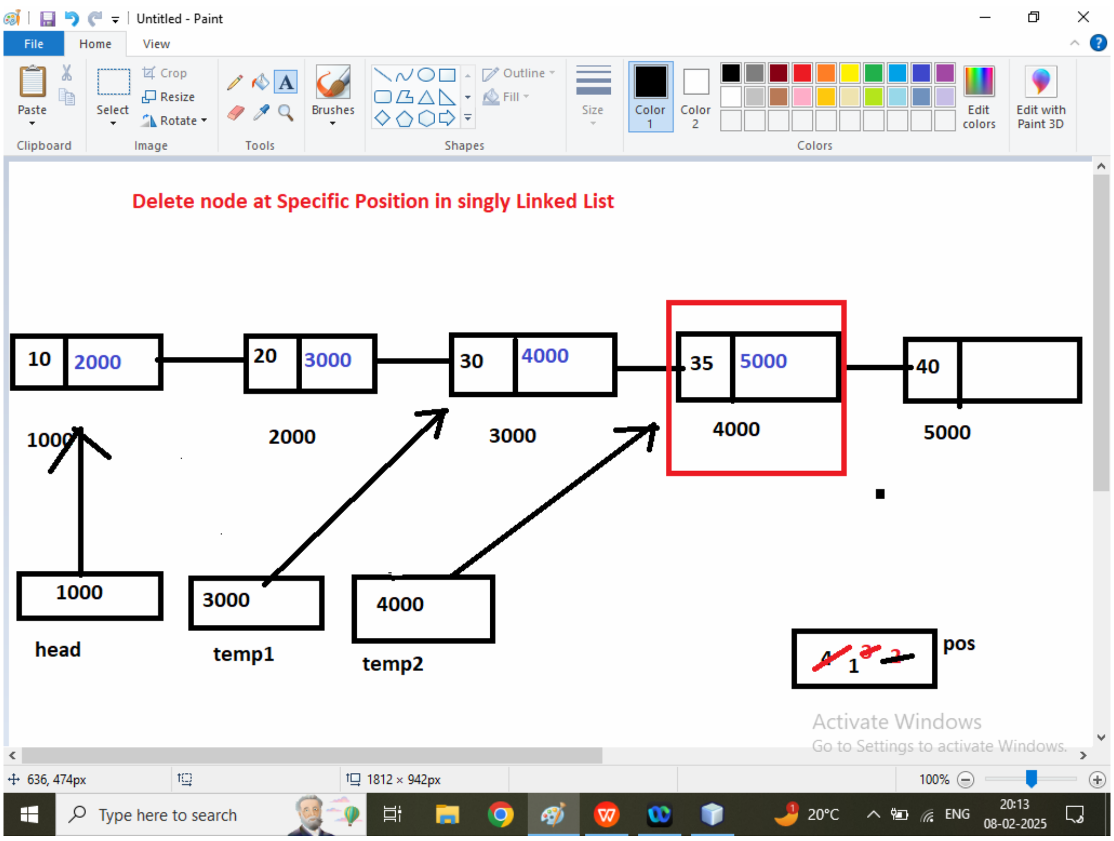


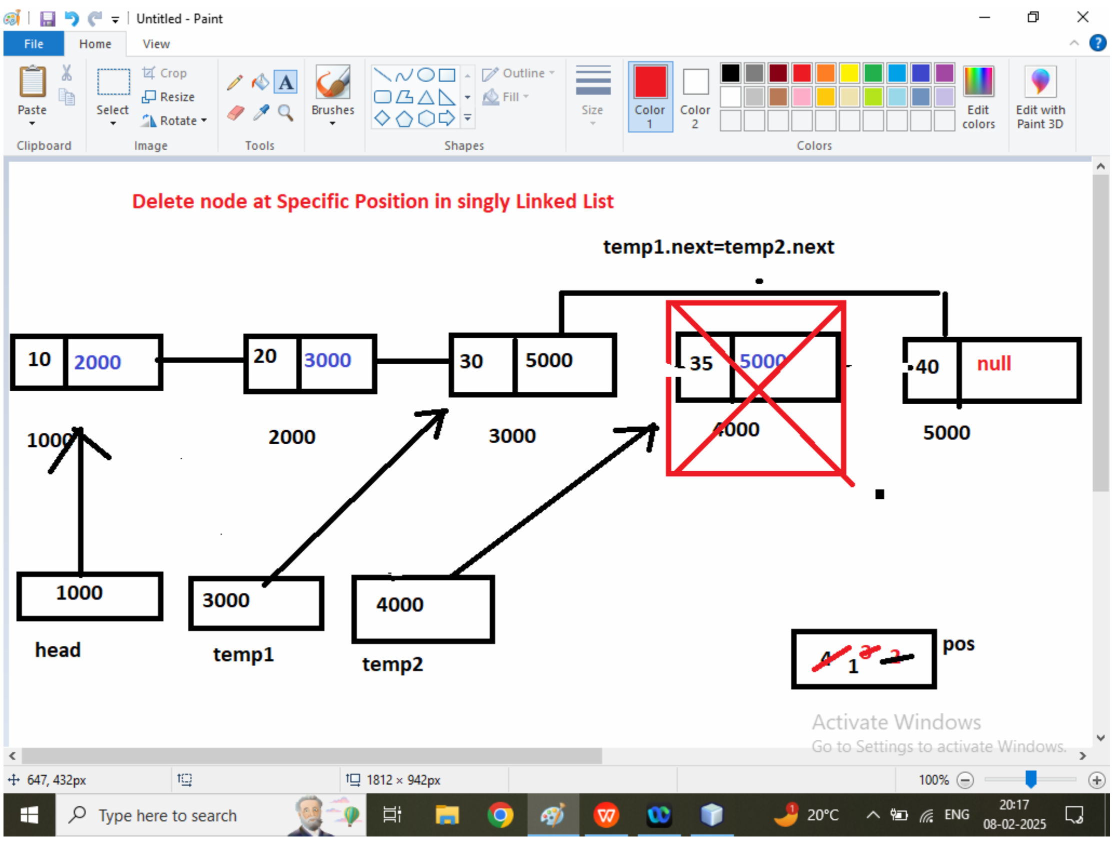
First Iteration




Delete node at Specific Position in singly Linked List

Second Iteration


Delete node at Specific Position in singly Linked List



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dsafeb2025;

/**
 *
 * @author Admin
 */
public class Node {

    int data;
    Node next;

    public Node(int data) {
        this.data = data;
        next = null;
        System.out.println("Node created Success");
    }
public void display(Node head){
    Node temp=head;
    while(temp!=null){
        System.out.print("---->"+temp.data);//10--->20--->30--->40
        temp=temp.next;
    }
}

public Node addNodeAtStart(Node head,int data){
   //step1: Create a new Node
   Node newNode=new Node(data);
   //step2: Make Point NewNode to Current Node
   newNode.next=head;
   //step3: Update head
   head=newNode;
   //step4: return new head

    return head;

}
public void addNewNodeAtEnd(Node head,int data){
    //step1:Create new Node
    Node newNode=new Node(data);
    //step2: Traverse the list
    Node temp=head;
    while(temp.next!=null){
        temp=temp.next;
    }
    //step3: Setting the last node next pointer to the new node
    temp.next=newNode;


}
public void addNewNodeAtPos(Node head,int data,int pos){
    //step1: Create a  new Node
    Node newNode=new Node(data);
    //step2: Traverse the list upto specified position
    pos--;//3
    Node temp1=head;
    Node temp2=head.next;
    while(pos>1){
        temp1=temp1.next;
        temp2=temp2.next;
        pos--;//1
    }
    temp1.next=newNode;
    newNode.next=temp2;

}
public boolean isEmpty(Node head){
    return head==null;
}

public void deleteLastNode(Node head){
    //step1: check list is empty
    if(isEmpty(head)){
        System.out.println("List is Empty");
    }else{
        //step2: Traverse The List Upto Last Node
        Node temp=head;
        while(temp.next.next!=null){
          temp=temp.next;
        }
        //step3: To Update temp.next=null
        temp.next=null;
    }
```

```
}
public Node deleteFirstNode(Node head){
   //step1: Check the list is empty
   if(isEmpty(head)){
      System.out.println("List is Empty");
   }else{
      //step2: Store head into temporary variable
      Node temp=head;
      //step3: Move head to the next node
      head=head.next;
      //step4: Free memory of the temporary variable
      temp=null;

   }
   return head;
}

public void deleteNodeAtPos(Node head,int pos){
   //step1: To Check List is Empty
   if(isEmpty(head)){
      System.out.println("List is Empty");
   }else{
      //step2: Traverse the list
      pos--;//3
      Node temp1=head;
      Node temp2=head.next;
      while(pos>1){
         temp1=temp1.next;
         temp2=temp2.next;
         pos--;//2
      }
      temp1.next=temp2.next;
   }
}
   public static void main(String[] args) {
      Node first = new Node(10);
      Node second = new Node(20);
      Node third = new Node(30);

      //Head point the first node of singly Linked List
      Node head=first;
      first.next=second;
      second.next=third;

      System.out.println("Print Data of Singly Linked List");
      System.out.println("===>"+first.data+"===>"+second.data+"===>"+third.data);
      System.out.println("Print Data of Singly Linked List Using head");
      System.out.print("===>"+head.data);
      System.out.print("===>"+head.next.data);
      System.out.print("===>"+head.next.next.data);

      System.out.println("Print Data of Singly Linked List Using Method");
      head.display(head);
     // head=head.addNodeAtStart(head, 5);
      System.out.println("\nPrint Data after New Node at starting in singly linked List");
      head.display(head);
      System.out.println("print Data After Add new Node at end of singly linked List");
      head.addNewNodeAtEnd(head, 40);
      head.display(head);
      System.out.println("\nPrint Data after insert new Node at Specific position\n");
      head.addNewNodeAtPos(head,35,4);
      head.display(head);
     // head=head.deleteFirstNode(head);
      System.out.println("\nPrint data of singly Linked List After deletion first Node \n");
      head.display(head);
    //   head.deleteLastNode(head);
      System.out.println("\nPrint data of singly Linked List After deletion Last Node \n");
      head.display(head);
      System.out.println("\nPrint data of singly Linked List After deletion  Node at Specific Position
\n");
      head.deleteNodeAtPos(head, 4);
      head.display(head);

   }
}
```

Q3. Explain Doubly Linked List in java?

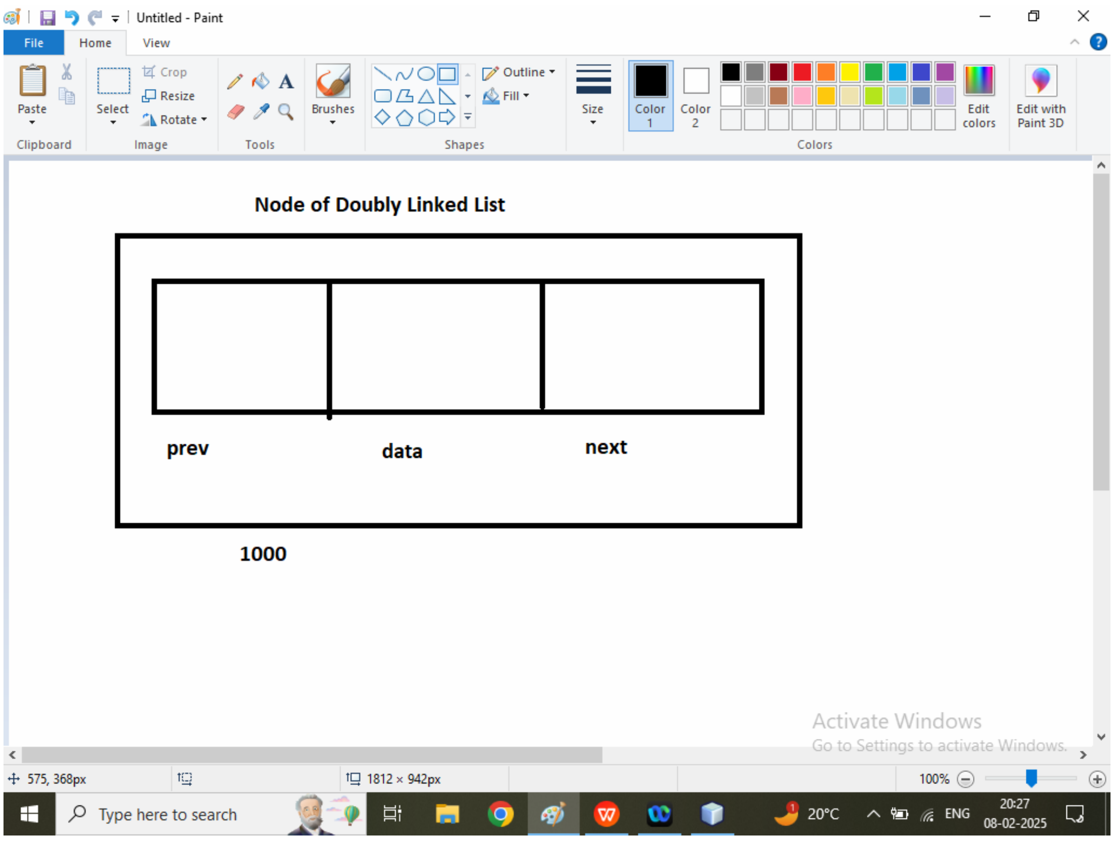
Ans: A Doubly Linked List is a type of linked List where each node contains

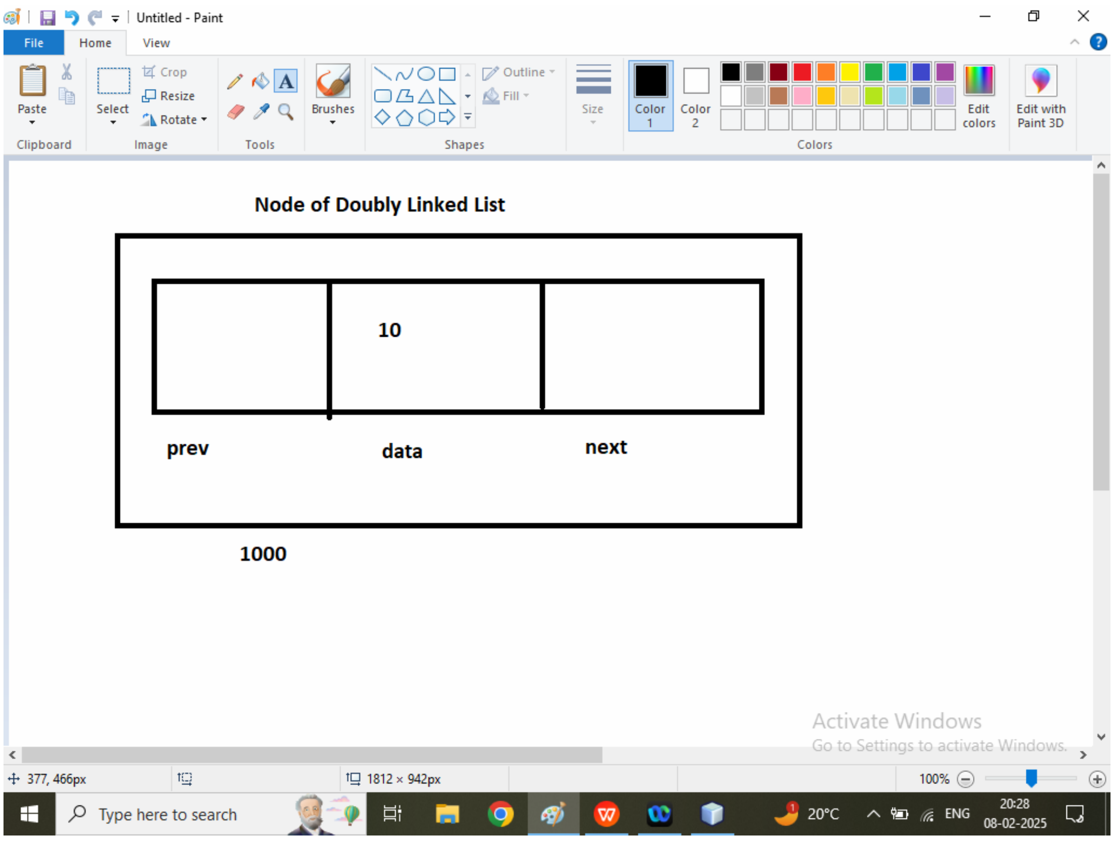
1. Data: Data
2. Next: Next node reference
3. Prev : previous node reference

The main advantage of doubly linked List is we can traverse the data in forward direction(with the next) and backward direction (with prev)

Operation of Doubly Linked List(Homework)

1. Insert new Node at starting of doubly Linked List
2. Insert new Node at specific position of doubly linked List
3. Insert new Node at the end of doubly Linked List
4. Delete First Node of doubly Linked List
5. Delete Last Node of Doubly Linked List
6. Delete Specific Node of Doubly Linked List





/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.

```
 */
package dsafeb2025;

/**
 *
 * @author Admin
 */
public class Node {
    Node prev;
    int data;
    Node next;

    public Node(int data){
        prev=null;
        this.data=data;
        next=null;
        System.out.println("Node Created In Doubly Linked List");
    }
    public static void main(String[] args) {
        Node f1=new Node(10);
    }
}
```