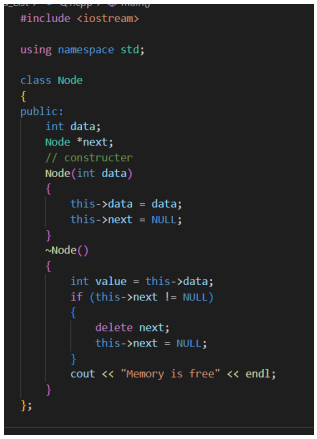
Implementation of Link list

Student Id: 202203031
Date : 12/04/2023

Q1:Write a program to implement linked list in CPP language.
CODE:



```cpp
#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;
    }
};
```

```cpp
// 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(12);
    cout << node1->data << endl;

    // head pointedd to node1
    Node *head = node1;
    Node *tail = node1;
    InsertAtHead(head, 24);
    print(head);

    InsertAtHead(head, 36);
    print(head);

    InsertAtTail(tail, 5);
    print(head);

    InsertAtTail(tail, 10);
    print(head);

    InsertAtPosition(tail, head, 3, 21);
    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;
```

```
    deleteNode(1, head);
    print(head);

    deleteNode(3, head);
    print(head);

    cout << "Head " << head->data << endl;
    cout << "Tail " << tail->data << endl;

    return 0;
}
```

```
PS C:\Users\parma\OneDrive\Desktop\Programming> cd "c:\Users\p
12
24 12
36 24 12
36 24 12 5
36 24 12 5 10
36 24 21 12 5 10
Head 36
Tail 10
Memory is free
24 21 12 5 10
Memory is free
24 21 5 10
Head 24
Tail 10
PS C:\Users\parma\OneDrive\Desktop\Programming\Linked_List>
```

Q2:Write a program to implement linked list in Java language.
CODE:

```java
class Linked_List {
    Node head;
    private int size;

    Linked_List(){
        this.size = 0;
    }

    class Node{
        String data;
        Node next;

    Node(String data){
        this.data = data;
        this.next = null;
        size++;
        }
    }

    //add = first , last

    public void addFirst(String data){
        Node newNode = new Node(data);

        if(head == null)
        {
                head = newNode;
                return;
        }
        newNode.next = head;
        head = newNode;
    }
```

```java
  // add - last
  public void addLast(String data){
      Node newNode = new Node(data);
      if(head == null){
          head = newNode;
          return;
      }

      Node curr = head;
      while(curr.next != null){
          curr = curr.next;
      }
      curr.next = newNode;
  }


  public void printList(){
      if(head == null){
          System.out.println("List is empty");
          return;
      }
      Node curr = head;
      while(curr != null){
          System.out.print(curr.data + " -> ");
          curr = curr.next;
      }
      System.out.println("NULL");

  }
```

```java
//delete first
public void deleteFirst(){
    if(head == null){
        System.out.println(x:"The list is empty");
        return;
    }
    size--;
    head = head.next;
}

//delete last
public void deleteLast(){
if(head == null){
        System.out.println(x:"The list is empty");
        return;
    }
size--;
if(head.next == null){
    head = null;
    return;
}
    Node secondLast = head;
    Node lastNode = head.next;
    while(lastNode.next != null){
        lastNode = lastNode.next;
        secondLast = secondLast.next;

    }
    secondLast.next = null;
}
```

```java
 public int getSize(){
 return size;
 }

 public void reverseIterate(){
    if(head == null || head.next == null){
        return;
    }

    Node prevNode = head;
    Node currNode = head.next;

    while(currNode != null){
        Node nextNode = currNode.next;
        currNode.next = prevNode;

        //update
        prevNode = currNode;
        currNode = nextNode;
    }
    head.next = null;
    head = prevNode;
 }

 public Node reverseReurcive(Node head){
    if(head == null || head.next == null){
        return head;
    }
    Node newHead = reverseReurcive(head.next);
    head.next.next = head;
    head.next = null;

    return newHead;
```

```java
}

Run | Debug
public static void main(String args[]){
    Linked_List list = new Linked_List()
    list.addFirst(data:"uday");
    list.printList();

    list.addFirst(data:"is");
    list.printList();

    list.addLast(data:"doing");
    list.printList();

    list.addLast(data:"B.tech ?");
    list.printList();

    list.deleteFirst();
    list.printList();
```

```java
    list.deleteLast();
    list.printList();

    System.out.println(list.getSize());
    list.addFirst(data:"Bachelor");
    System.out.println(list.getSize());

    list.printList();
    list.reverseIterate();
    list.printList();

    list.printList();
    list.head = list.reverseReurcive(list.head);
    list.printList();
}
```

OUTPUT:

```
uday -> NULL
is -> uday -> NULL
is -> uday -> doing -> NULL
is -> uday -> doing -> B.tech ? -> NULL
uday -> doing -> B.tech ? -> NULL
uday -> doing -> NULL
2
3
Bachelor -> uday -> doing -> NULL
doing -> uday -> Bachelor -> NULL
doing -> uday -> Bachelor -> NULL
Bachelor -> uday -> doing -> NULL
PS C:\Users\parma\OneDrive\Desktop\Programming>
```

Q3:Make a Linked List & add the following elements to it : (1, 5, 7, 3 , 8, 2, 3). Search for the number 7 & display its index.

CODE:

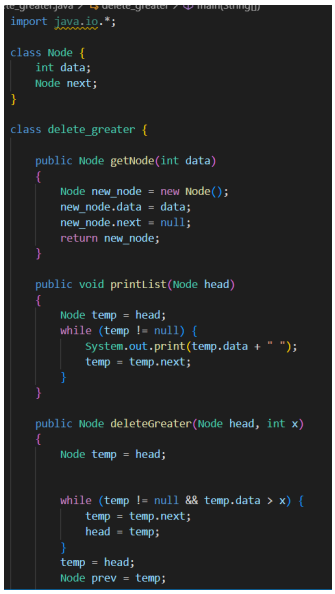
```java
class LinkedList {
  Node head;

  class Node {
    int data;
    Node next;

    Node(int x) {
      data = x;
      next = null;
    }
  }

  public void search_Node(int data) {
    Node current = head;
    int i = 1;
    boolean flag = false;

    if (head == null) {
      System.out.println(x:"List is empty");
    } else {
      while (current != null) {
        if (current.data == data) {
          flag = true;
          break;
        }
        i++;
        current = current.next;
      }
    }
    if (flag)
      System.out.println("element is at index " + i);
    else
      System.out.println(x:"Element is not  in the list");
  }
```

```java
  public Node insert(int data) {
    Node newNode = new Node(data);
    newNode.next = head;
    head = newNode;

    return head;
  }

  public void display() {
    Node node = head;
    while (node != null) {
      System.out.print(node.data + " ");
      node = node.next;
    }
    System.out.println(x:"\n");
  }
}

public class LL {
  Run | Debug
  public static void main(String args[]) {
    LinkedList ll = new LinkedList();

    ll.insert(data:3);
    ll.insert(data:2);
    ll.insert(data:8);
    ll.insert(data:3);
    ll.insert(data:7);
    ll.insert(data:5);
    ll.insert(data:1);


    ll.search_Node(data:7);
  }
}
```

OUTPUT:

```
element is at index 3
PS C:\Users\parma\OneDrive\Desktop\Programming\Linked_list(java)>
```

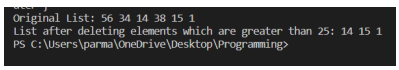
Q4:Take elements(numbers in the range of 1-50) of a Linked List as input from the user. Delete all nodes which have values greater than 25.



```java
import java.io.*;

class Node {
    int data;
    Node next;
}

class delete_greater {

    public Node getNode(int data)
    {
        Node new_node = new Node();
        new_node.data = data;
        new_node.next = null;
        return new_node;
    }

    public void printList(Node head)
    {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
    }

    public Node deleteGreater(Node head, int x)
    {
        Node temp = head;


        while (temp != null && temp.data > x) {
            temp = temp.next;
            head = temp;
        }
        temp = head;
        Node prev = temp;
```

```java
        while (temp != null) {
            while (temp != null && temp.data <= x) {
                prev = temp;
                temp = temp.next;
            }
            if (temp == null) {
                return head;
            }
            prev.next = temp.next;
            temp = prev.next;
        }
        return head;
    }


    Run | Debug
    public static void main(String[] args)
    {

        delete_greater list = new delete_greater();

        Node head = list.getNode(data:56);
        head.next = list.getNode(data:34);
        head.next.next = list.getNode(data:14);
        head.next.next.next = list.getNode(data:38);
        head.next.next.next.next = list.getNode(data:15);
        head.next.next.next.next.next = list.getNode(data:1);

        System.out.print(s:"Original List: ");
        list.printList(head);

        int x = 25;
        head = list.deleteGreater(head,x);
        System.out.print("\nList after deleting elements which are greater than"+x+": ");
        list.printList(head);
    }
}
```

OUTPUT:



```
Original List: 56 34 14 38 15 1
List after deleting elements which are greater than 25: 14 15 1
PS C:\Users\parma\OneDrive\Desktop\Programming>
```