Recitation Problem: Linked List

CS 2003 – Data Structures

Spring 2023

Overview

In this recitation problem, students will review the following topics:

1. Singly Linked List
2. Doubly Linked List

Problem 1 Create an algorithm that when given the head of a singly linked list it will delete every other node.

```
public void deleteEveryOther(Node head){
    // write your code here
}
```

Problem 2 Given a singly length list, create an algorithm that will return the length of that list.

```
public int listLength(Node head){
    // write your code here
}
```

Problem 3 Given a singly linked list and a value, create a search algorithm that will return true if the value exists in the list and return false if it does not exist in the list.

```
public Node removeElements(Node head , int val){
    //write your search solution here
}
```

Problem 4 Create a recursive algorithm that will remove all nodes with value K from a singly linked list.**Your creating a new list**

```
public Node removeElements(Node head , int val){
    //write your recursion solution here
}
```

Problem 5 Given a singly linked list, create an algorithm that will reverse that linked list and return its new head.

```
public Node reverseLinkedList(Node head){
    //write your recursion solution here
}
```

Problem 6 Create a recursive algorithm that will Merge two sorted singly linked lists into one sorted list. Assume that the heads of two sorted list are passed into your method.(*Use the Node class above*)

```
public Node mergeTwoLists(Node list1 , Node list2){
    //write your recursion solution here
}
```

Problem 7 Given two Unsorted Doubly Linked Lists, merge them together into one Linked List.

```
public Node mergeTwoUnsortedLists(Node list1 , Node list2){
    //write your merge solution here
}
```

Problem 8 A helper Node class is provided to you. For the following list, Create an insert algorithm that inserts into a sorted doubly linked list.

```
public class Node{
    int val;
    Node next;
    Node prev;

    public Node(){

    }

    public Node(int val){
        this.val = val;
    }
}

public class myLinkedList{
    Node head;

    public void insert(Node x){

        \\write your insert algorithm here
    }

}
```

Problem 9 Given the following code, trace the linked list and determine the output.

```
public class Node{
    public int val;
    public Node next;
    public static Node prev;

    public Node(int val){
        this.val = val;
        this.next = null;
    }

    public static void main(String[] args){
        Node a = new Node(7);
        Node b = new Node(19);
        Node c = new Node(26);
        Node d = new Node(5);
        Node e = new Node(11);

        a.next = b;
        b.next = c;
        c.next = d;
        d.next = e;
        e.next = a;

        a.prev = e;
        b.prev = a;
        c.prev = b;
        d.prev = c;
        e.prev = d;

        d.prev.next.prev = c.next.next.next.prev.next;
        a.next.next.next.next = b.prev.next.prev.prev.next.next;
        e.prev.next.next = c.prev.prev.prev.next.next.next;
        b.prev.prev.next.prev.next = d.next.next.next.prev.next.next.next;
        System.out.print(a.next.next.prev.prev.next.next.next.val);
    }
}
```

Problem 10 Given a sorted Doubly Linked List, remove all duplicates from that list.

```
public Node removeDuplicatesFromDLL(Node head){
    //write your remove solution here
}
```

Problem 11 A helper Node class is provided to you.For the following list, Create an insert algorithm that only inserts at the front of the doubly linked list.

```
public class Node{
    int val;
    Node next;
    Node prev;

    public Node(){

    }

    public Node(int val){
        this.val = val;
    }

    public Node(int val, Node next){
        this.val = val;
        this.next = next;
    }
}

public class myLinkedList{
    Node head;

    public void insert(Node x){
        \\write your insert algorithm here
    }
}
```

Problem 12 A helper Node class is provided to you.For the following list, Create an insert algorithm that only inserts at the end of the doubly linked list.

```
public class Node{
    int val;
    Node next;
    Node prev;

    public Node(){

    }

    public Node(int val){
        this.val = val;
    }

    public Node(int val, Node next){
     this.val = val;
     this.next = next;
```

```
    }
}

public class myLinkedList{
    Node head;

    public void insert(Node x){
        \\write your insert algorithm here
    }
}
```