Assignment Questions 12

Question 1

Given a singly linked list, delete **middle** of the linked list. For example, if given linked list is 1->2->**3**->4->5 then linked list should be modified to 1->2->4->5.If there are **even** nodes, then there would be **two middle** nodes, we need to delete the second middle element. For example, if given linked list is 1->2->3->4->5->6 then it should be modified to 1->2->3->5->6.If the input linked list is NULL or has 1 node, then it should return NULL

Example 1:

Input:

LinkedList: 2->4->6->7->5->1

Output:2 4 6 5 1

Solution:-

```
import java.io.*;


class Test {
	static class Node {
		int data;
		Node next;
	}
	static Node newNode(int data)
	{
		Node temp = new Node();
		temp.data = data;
		temp.next = null;
		return temp;
	}
	static int countOfNodes(Node head)
	{
```

```
        int count = 0;
        while (head != null) {
                head = head.next;
                count++;
        }
        return count;
}
static Node deleteMid(Node head)
{

        if (head == null)
                return null;
        if (head.next == null) {
                return null;
        }
        Node copyHead = head;

        int count = countOfNodes(head);
        int mid = count / 2;

        // Delete the middle node
        while (mid-- > 1) {
                head = head.next;
        }

        // Delete the middle node
        head.next = head.next.next;
```

```
        return copyHead;
}

// A utility function to print
// a given linked list
static void printList(Node ptr)
{
        while (ptr != null) {
                System.out.print(ptr.data + "->");
                ptr = ptr.next;
        }
        System.out.println("NULL");
}
public static void main(String[] args)
{
        /* Start with the empty list */
        Node head = newNode(1);
        head.next = newNode(2);
        head.next.next = newNode(3);
        head.next.next.next = newNode(4);

        System.out.println("Given Linked List");
        printList(head);

        head = deleteMid(head);
```

```
                System.out.println(
                        "Linked List after deletion of middle");
                printList(head);
        }
}
```

Question 2

Given a linked list of **N** nodes. The task is to check if the linked list has a loop. Linked list can contain self loop.

Example 1:

Input:

N = 3

value[] = {1,3,4}

x(position at which tail is connected) = 2

Output:True

Explanation:In above test case N = 3.

The linked list with nodes N = 3 is

given. Then value of x=2 is given which

means last node is connected with xth

node of linked list. Therefore, there

exists a loop.

Solution:-

```
    class Solution
{
  public static boolean detectLoop(Node head)
  {
     HashSet<Node> hs = new HashSet<>();
```

```
    for(Node curr = head;curr != null;curr = curr.next)
    {
      if(hs.contains(curr))
        return true;
      hs.add(curr);
    }
    return false;
  }
}
```

Question 3

Given a linked list consisting of **L** nodes and given a number **N**. The task is to find the **N**th node from the end of the linked list.

Example 1:

Input:

N = 2

LinkedList: 1->2->3->4->5->6->7->8->9

Output:8

Explanation:In the first example, there

are 9 nodes in linked list and we need

to find 2nd node from end. 2nd node

from end is 8.

Solution:-

```
class Solution
{
  //Function to find the data of nth node from the end of a linked list.
```

```
int getNthFromLast(Node head, int n)
    {
        if(head == null || n<=0){
            return -1;
        }
        Node h1 = head;
        Node h2 = head;
        for(int i=0;i<n-1;i++){
            if(h1 == null){
                return -1;
            }
            h1 = h1.next;
        }

        if(h1 == null){
            return -1;
        }

        while(h1.next != null){
            h1 = h1.next;
            h2 = h2.next;
        }
        return h2.data;
```

```
    }
}
```

Question 4

Given a singly linked list of characters, write a function that returns true if the given list is a palindrome, else false.

Examples:

Input: R->A->D->A->R->NULL

Output: Yes

Input: C->O->D->E->NULL

Output: No

Solution:-

```
import java.util.*;

class linkedList {

        public static void main(String args[])

        {

                Node one = new Node(1);

                Node two = new Node(2);

                Node three = new Node(3);

                Node four = new Node(4);

                Node five = new Node(3);

                Node six = new Node(2);

                Node seven = new Node(1);

                one.ptr = two;
```

```
        two.ptr = three;
        three.ptr = four;
        four.ptr = five;
        five.ptr = six;
        six.ptr = seven;
        boolean condition = isPalindrome(one);
        System.out.println("isPalidrome :" + condition);
}
static boolean isPalindrome(Node head)
{

        Node slow = head;
        boolean ispalin = true;
        Stack<Integer> stack = new Stack<Integer>();

        while (slow != null) {
                stack.push(slow.data);
                slow = slow.ptr;
        }

        while (head != null) {

                int i = stack.pop();
```

```
                    if (head.data == i) {
                        ispalin = true;
                    }
                    else {
                        ispalin = false;
                        break;
                    }
                    head = head.ptr;
                }
                return ispalin;
        }
}

class Node {
        int data;
        Node ptr;
        Node(int d)
        {
                ptr = null;
                data = d;
        }
}
```

<aside> 💡 **Question 5**

Given a linked list of **N** nodes such that it may contain a loop.

A loop here means that the last node of the link list is connected to the node at position X(1-based index). If the link list does not have any loop, X=0.

Remove the loop from the linked list, if it is present, i.e. unlink the last node which is forming the loop.

Example 1:

Input:

N = 3

value[] = {1,3,4}

X = 2

Output:1

Explanation:The link list looks like

1 -> 3 -> 4

^ |

|____|

A loop is present. If you remove it

successfully, the answer will be 1.

Solution:-

```
class Solution
{
    public static void removeLoop(Node head){
      Set<Node> s=new HashSet<>();
      Node curr=head;
      while(curr!=null){
```

```
        if(s.contains(curr.next)){
            curr.next=null;
            return;
        }
        s.add(curr);
        curr=curr.next;
      }


   }
}
```

Question 6

Given a linked list and two integers M and N. Traverse the linked list such that you retain M nodes then delete next N nodes, continue the same till end of the linked list.

Difficulty Level: Rookie

Input:

M = 2, N = 2

Linked List: 1->2->3->4->5->6->7->8

Output:

Linked List: 1->2->5->6

Input:

M = 3, N = 2

Linked List: 1->2->3->4->5->6->7->8->9->10

Output:

Linked List: 1->2->3->6->7->8

Input:

M = 1, N = 1

Linked List: 1->2->3->4->5->6->7->8->9->10

Output:

Linked List: 1->3->5->7->9

Solution:-

```
class Solution
{
   static void linkdelete(Node head, int M, int N)
   {
      Node temp1=head;
      Node temp=temp1;

      while(temp1!=null){
      temp=temp1;
      int i=1;
      while(temp!=null&&i<M) {
         temp=temp.next;
         i++;
      }
```

```
        if(temp==null) return;
        i=1;
        while(temp1!=null&&i<=(M+N)){
            temp1=temp1.next;
            i++;
        }
        temp.next=temp1;
        }
    }
}
```

Question 7

Given two linked lists, insert nodes of second list into first list at alternate positions of first list. For example, if first list is 5->7->17->13->11 and second is 12->10->2->4->6, the first list should become 5->12->7->10->17->2->13->4->11->6 and second list should become empty. The nodes of second list should only be inserted when there are positions available. For example, if the first list is 1->2->3 and second list is 4->5->6->7->8, then first list should become 1->4->2->5->3->6 and second list to 7->8.

Use of extra space is not allowed (Not allowed to create additional nodes), i.e., insertion must be done in-place. Expected time complexity is O(n) where n is number of nodes in first list.

Solution:-

```
class LinkedList
{
        Node head; // head of list
        class Node
        {
                int data;
```

```
        Node next;
        Node(int d) {data = d; next = null; }
    }

    /* Inserts a new Node at front of the list. */
    void push(int new_data)
    {
        Node new_node = new Node(new_data);
        new_node.next = head;
        head = new_node;
    }
.
    void merge(LinkedList q)
    {
        Node p_curr = head, q_curr = q.head;
        Node p_next, q_next;
        while (p_curr != null && q_curr != null) {

            // Save next pointers
            p_next = p_curr.next;
            q_next = q_curr.next;

            q_curr.next = p_next; // change next pointer of q_curr
```

```
                p_curr.next = q_curr; // change next pointer of p_curr
                p_curr = p_next;
                q_curr = q_next;
        }
        q.head = q_curr;
}
void printList()
{
        Node temp = head;
        while (temp != null)
        {
        System.out.print(temp.data+" ");
        temp = temp.next;
        }
        System.out.println();
}
public static void main(String args[])
{
        LinkedList llist1 = new LinkedList();
        LinkedList llist2 = new LinkedList();
        llist1.push(3);
        llist1.push(2);
        llist1.push(1);
```

```
        System.out.println("First Linked List:");
        llist1.printList();

        llist2.push(8);
        llist2.push(7);
        llist2.push(6);
        llist2.push(5);
        llist2.push(4);

        System.out.println("Second Linked List:");

        llist1.merge(llist2);

        System.out.println("Modified first linked list:");
        llist1.printList();

        System.out.println("Modified second linked list:");
        llist2.printList();
    }
}
```

Question 8

Given a singly linked list, find if the linked list is circular or not.

A linked list is called circular if it is not NULL-terminated and all nodes are connected in the form of a cycle. Below is an example of a circular linked list.

```
import java.util.*;


class Test {

	static class Node {

		int data;

		Node next;

	}


	static boolean isCircular(Node head)

	{

		// An empty linked list is circular

		if (head == null)

			return true;

		Node node = head.next;

		while (node != null && node != head)

			node = node.next;

		return (node == head);

	}


		static Node newNode(int data)

	{
```

```
        Node temp = new Node();
        temp.data = data;
        temp.next = null;
        return temp;
    }
    public static void main(String args[])
    {
        /* Start with the empty list */
        Node head = newNode(1);
        head.next = newNode(2);
        head.next.next = newNode(3);
        head.next.next.next = newNode(4);

        System.out.print(isCircular(head) ? "Yes: "No\n");

        // Making linked list circular
        head.next.next.next.next = head;

        System.out.print(isCircular(head) ? "Yes\n": "No\n");
    }
}
```