```java
// Question 2 : Write a program to create a singly linked list and do the
following
    // i. Find the average of middle 2 elements
    // ii. Delete last node
    // iii. Given 2 positions, swap the values at the positions
// Author : Sharun E Rajeev

import java.io.*;

public class LabB2 {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        int choice;
        LinklistFunctions lf = new LinklistFunctions();
        do {
            System.out.println("\nLinked List Manager");
            System.out.println("1.Add Data to Linked List.");
            System.out.println("2.Display Data in the Linked List.");
            System.out.println("3.Average of Middle Elements.");
            System.out.println("4.Delete last node.");
            System.out.println("5.Swap values.");
            System.out.println("6.Exit.");
            System.out.println("Enter your choice.");
            choice = Integer.parseInt(br.readLine());
            switch (choice) {
            case 1:
                System.out.println("Enter data to be inserted.");
                int data = Integer.parseInt(br.readLine());
                lf.push(data);
                lf.displayLinkedList();
                break;
            case 2:
                lf.displayLinkedList();
                break;
            case 3:
                lf.averageOfMiddle();
                break;
            case 4:
                lf.deleteLastNode();
                lf.displayLinkedList();
                break;
            case 5:
                System.out.println("Enter 2 position to be swapped.");
                String line = br.readLine();
                String[] str = line.trim().split(" ");
                int p1 = Integer.parseInt(str[0]);
                int p2 = Integer.parseInt(str[1]);
                if (p1 == p2)
                    System.out.println("Swap not possible, same postion
specified.");
                else
                    lf.swap(p1, p2);
                lf.displayLinkedList();
                break;
            case 6:
                System.out.println("Have a great day!");
                break;
            default:
                System.out.println("Wrong option selected. ");
            }
        } while (choice != 6);
    }
}

class Link {
    private int data;
    public Link next;

    public Link(int d) {
        data = d;
    }

    public int returnData() {
        return data;
    }

    public void setData(int d) {
        data = d;
    }
}

class LinklistFunctions {
    private Link first;

    public LinklistFunctions() {
        first = null;
    }

    public boolean isEmpty() {
        return first == null;
    }

    // Insert data to the last of the linked list
    public Link push(int data) {
        Link li = null;
        Link new_link = new Link(data);
        new_link.next = null;
        if (first == null) {
            first = new_link;
        } else {
            Link last = first;
            while (last.next != null) {
                last = last.next;
            }
            last.next = new_link;
        }
        return li;
    }

    // Display the LinkedList
    public void displayLinkedList() {
        System.out.println("\nDisplaying Linked List");
        Link current = first;
        while (current != null) {
            System.out.print(current.returnData() + " ");
            current = current.next;
        }
        System.out.println(" ");
    }

    public int linkedListLength() { // Finds the length of the list.
        int count = 0;
        Link current = first;
        while (current != null) {
            count += 1;
            current = current.next;
        }
        return count;
    }

    public void averageOfMiddle() { // Finds the average of the middle elements
        Link current = first;
        int n = linkedListLength();
        if (n == 1) {
            System.out.println("Average = " + (current.returnData())/2);
        }
        else if (n % 2 == 0) {
            for (int i = 0; i < (n / 2) - 1; i++) {
                current = current.next;
            }
            int b = current.returnData();
            current = current.next;
            int c = current.returnData();
            System.out.println("Average of 2 middle elements = " + ((b + c) /
2));
        } else {
            for (int i = 1; i <= (n / 2); i++) {
                current = current.next;
            }
            int v = current.returnData();
            System.out.println("Average of the middle element = " + v / 2);
        }

    }

    public Link deleteLastNode() { // Delete the last node from linked list
        if (first == null)
            return null;
        if (first.next == null) {
            return null;
        }
        Link prev = first;
        while (prev.next.next != null)
            prev = prev.next;
        prev.next = null;
        return first;
    }

    public void swap(int pos1, int pos2) { // pos1 < pos2
        Link ithlink = first;
        Link jthlink = first;
        for (int link = 0; link < pos1-1; link++) {
            ithlink = ithlink.next;
        }
        for (int link = 0; link < pos2-1; link++) {
            jthlink = jthlink.next;
        }
        int temp = ithlink.returnData();
        ithlink.setData(jthlink.returnData());
        jthlink.setData(temp);
    }
}
```