7. Write a menu driven program to create a linked list and perform the following operations.

a. to Insert some Elements at the Specified Position

b. swap two elements in a linked list

c. to Iterate a LinkedList in Reverse Order

d. to Compare Two LinkedList

e. to Convert a LinkedList to ArrayList

```
import java.util.*;

public class LList {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinkedList<Integer> linkedList = new LinkedList<>();
        int choice;

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Create linked list");
            System.out.println("2. Display linked list");
            System.out.println("3. Insert elements at specified position");
            System.out.println("4. Swap two elements");
            System.out.println("5. Iterate the linked list in reverse order");
            System.out.println("6. Compare two linked lists");
            System.out.println("7. Convert linked list to array list");
            System.out.println("8. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    createLinkedList(linkedList, scanner);
                    break;
                case 2:
                    displayLinkedList(linkedList);
                    break;
                case 3:
                    insertElementAtPosition(linkedList, scanner);
                    break;
                case 4:
                    swapElements(linkedList, scanner);
                    break;
                case 5:
                    iterateReverseOrder(linkedList);
                    break;
                case 6:
                    compareLinkedLists(linkedList, scanner);
                    break;
                case 7:
                    convertToArrayList(linkedList);
                    break;
                case 8:
                    System.out.println("Exiting program...");
                    break;
                default:
                    System.out.println("Invalid choice! Please enter a valid option.");
            }
        } while (choice != 8);

        scanner.close();
    }

    private static void createLinkedList(LinkedList<Integer> linkedList, Scanner scanner) {
        linkedList.clear();
        System.out.print("Enter the number of elements: ");
        int n = scanner.nextInt();
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            linkedList.add(scanner.nextInt());
        }
        System.out.println("Linked list created successfully!");
    }

    private static void displayLinkedList(LinkedList<Integer> linkedList) {
        if (linkedList.isEmpty()) {
            System.out.println("Linked list is empty.");
        } else {
            System.out.println("Linked list:");
            for (Integer element : linkedList) {
                System.out.print(element + " ");
            }
            System.out.println();
        }
    }

    private static void insertElementAtPosition(LinkedList<Integer> linkedList, Scanner
scanner) {
        if (linkedList.isEmpty()) {
            System.out.println("Linked list is empty. Please create the linked list first.");
            return;
        }
        System.out.print("Enter the element to insert: ");
        int element = scanner.nextInt();
        System.out.print("Enter the position to insert at: ");
        int position = scanner.nextInt();
        linkedList.add(position, element);
        System.out.println("Element inserted successfully!");
    }

    private static void swapElements(LinkedList<Integer> linkedList, Scanner scanner) {
        if (linkedList.isEmpty()) {
            System.out.println("Linked list is empty. Please create the linked list first.");
            return;
        }
        System.out.print("Enter the first element to swap: ");
        int firstIndex = scanner.nextInt();
        System.out.print("Enter the second element to swap: ");
        int secondIndex = scanner.nextInt();
        Collections.swap(linkedList, firstIndex, secondIndex);
        System.out.println("Elements swapped successfully!");
    }

    private static void iterateReverseOrder(LinkedList<Integer> linkedList) {
        if (linkedList.isEmpty()) {
            System.out.println("Linked list is empty. Please create the linked list first.");
            return;
        }
        System.out.println("Linked list in reverse order:");
        ListIterator<Integer> iterator = linkedList.listIterator(linkedList.size());
        while (iterator.hasPrevious()) {
            System.out.print(iterator.previous() + " ");
        }
        System.out.println();
    }

    private static void compareLinkedLists(LinkedList<Integer> linkedList, Scanner scanner)
{
        if (linkedList.isEmpty()) {
            System.out.println("Linked list is empty. Please create the linked list first.");
            return;
        }
        LinkedList<Integer> secondList = new LinkedList<>(linkedList);
        System.out.println("Enter number of elements for second linked list:");
        int count = scanner.nextInt();
         System.out.println("Enter elements :");
        for (int i = 0; i < count; i++) {
            secondList.add(scanner.nextInt());
        }
        boolean isEqual = linkedList.equals(secondList);
        if (isEqual) {
            System.out.println("The two linked lists are equal.");
        } else {
            System.out.println("The two linked lists are not equal.");
        }
    }

    private static void convertToArrayList(LinkedList<Integer> linkedList) {
        if (linkedList.isEmpty()) {
            System.out.println("Linked list is empty. Please create the linked list first.");
            return;
        }
        ArrayList<Integer> arrayList = new ArrayList<>(linkedList);
        System.out.println("Linked list converted to array list: " + arrayList);
    }
}
```