Backtracking Assignment Questions

Q1. Given a linked list and a key 'X' in, the task is to check if X is present in the linked list or not.

Examples: Input: 14->21->11->30->10, X = 14
Output: Yes
Explanation: 14 is present in the linked list.
Input: 6->21->17->30->10->8, X = 13
Output: No

Ans :-

```
package LinkedListAssignment;

import java.util.*;

public class QuestionOne {
    Node head;

    class Node {
        int data;
        Node next;

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

    public void insertAtEnd(int newData) {
        Node newNode = new Node(newData);
        if (head == null) {
            head = newNode;
            return;
        }

        newNode.next = null;
        Node temp = head;

        while (temp.next != null) {
            temp = temp.next;
        }

        temp.next = newNode;

    }

    public void check(int x) {
        Node current = head;
        boolean found = false;
        while (current.next != null) {

            if (current.data == x) {
                found = true;
                System.out.println("yes");
                break;
            }
            current = current.next;
        }

        if (!found) {
            System.out.println("No");
        }

    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        QuestionOne q1 = new QuestionOne();
        System.out.println("Enter the number of digits present in the
linked list");
        int p = sc.nextInt();
        System.out.println("Enter the digits present in the linked list");
        for (int i = 0; i < p; i++) {
            int digit = sc.nextInt();
            q1.insertAtEnd(digit);
        }

        System.out.println("Enter the value x");
        int x = sc.nextInt();
        q1.check(x);
    }
}
```

Output :-

```
Enter the number of digits present in the linked list
5
Enter the digits present in the linked list
14 21 11 30 10
Enter the value x
14
```

yes

Q2. Insert a node at the given position in a linked list. We are given a pointer to a node, and the new node is inserted after the given node.
Input: LL = 1-2-4-5-6 pointer = 2 value = 3.
Output: 1-2-3-4-5-6

Ans :-

```java
package LinkedListAssignment;

import java.util.Scanner;

public class QuestionTwo {
    Node head;

    class Node {
        int data;
        Node next;

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

    public void insertAtEnd(int newData) {
        Node newNode = new Node(newData);
        if (head == null) {
            head = newNode;
            return;
        }

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

    public void insertAtPoint(Node pointer, int val) {
        if (pointer == null) {
            System.out.println("Invalid pointer: Cannot insert at the
given position.");
            return;
        }
        Node newNode = new Node(val);
        newNode.next = pointer.next;
        pointer.next = newNode;
    }

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

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        QuestionTwo q1 = new QuestionTwo();

        System.out.println("Enter the number of digits present in the
linked list:");
        int p = sc.nextInt();

        System.out.println("Enter the digits present in the linked
list:");
        for (int i = 0; i < p; i++) {
            int digit = sc.nextInt();
            q1.insertAtEnd(digit);
```

```java
        }

        System.out.println("Enter the pointer position :");
        int pointer = sc.nextInt() - 1;

        Node r = q1.head;
        for (int i = 0; i < pointer; i++) {
            if (r == null) {
                System.out.println("Invalid pointer: Position is out of bounds.");
                return;
            }
            r = r.next;
        }

        System.out.println("Enter the value to insert:");
        int value = sc.nextInt();

        q1.insertAtPoint(r, value);
        q1.printNode();
    }
}
```

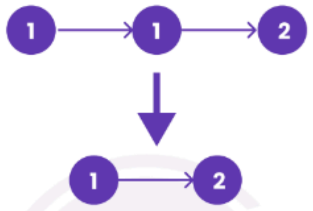
Output :-

```
Enter the number of digits present in the linked list:
5
Enter the digits present in the linked list:
1 2 4 5 6
Enter the pointer position :
2
Enter the value to insert:
3
1 2 3 4 5 6
```

Q3

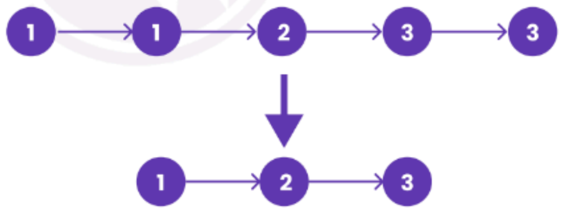
Q3. Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list sorted as well.

Example 1:


$$\begin{tikzcd} 1 \arrow[r] & 1 \arrow[r] \arrow[d] & 2 \\ & 1 \arrow[r] & 2 \end{tikzcd}$$

Input: head = [1,1,2]
Output: [1,2]

Example 2:


$$\begin{tikzcd} 1 \arrow[r] & 1 \arrow[r] & 2 \arrow[r] \arrow[d] & 3 \arrow[r] & 3 \\ & 1 \arrow[r] & 2 \arrow[r] & 3 & \end{tikzcd}$$

Input: head = [1,1,2,3,3]
Output: [1,2,3]

Ans :-

```
package LinkedListAssignment;

import java.util.*;

public class QuestionThree {
    Node head;

    class Node {
        int data;
        Node next;

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

```java
        }
    }

    public void insertAtEnd(int newData) {
        Node newNode = new Node(newData);
        if (head == null) {
            head = newNode;
            return;
        }

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

    public void removeDuplicates() {
        if (head == null)
            return;

        Set<Integer> seen = new HashSet<>();
        Node current = head;
        Node prev = null;

        while (current != null) {
            if (seen.contains(current.data)) {
                // Duplicate found; skip the current node
                prev.next = current.next;
            } else {
                seen.add(current.data);
                prev = current;
            }
            current = current.next;
        }
    }
    public void printNode() {
        Node curr = head;
        while (curr != null) {
            System.out.print(curr.data + " ");
            curr = curr.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        QuestionThree q1 = new QuestionThree();

        System.out.println("Enter the number of elements in the linked
list:");
        int length = sc.nextInt();

        System.out.println("Enter the elements of the linked list:");
        for (int i = 0; i < length; i++) {
            int digit = sc.nextInt();
            q1.insertAtEnd(digit);
        }

        q1.removeDuplicates();
        System.out.println("Linked list after removing duplicates:");
        q1.printNode();

        sc.close();
    }
}
```

Output :-

```
Enter the number of elements in the linked list:
3
Enter the elements of the linked list:
```

1 1 2
Linked list after removing duplicates:
1 2

Q4. Given the head of a singly linked list, return true if it is a palindrome or false otherwise.

Example 1: Input: head = [1,2,2,1]
Output: true
Example 2: Input: head = [1,2]
Output: false

Ans :-

```
package LinkedListAssignment;

import java.util.*;

public class QuestionFour {
    Node head;

    class Node {
        int data;
        Node next;

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

    public void insertAtEnd(int newData) {
        Node newNode = new Node(newData);
        if (head == null) {
            head = newNode;
```

```java
            return;
        }

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

    public boolean isPalindrome() {
        List<Integer> arlist = new ArrayList<>();

        Node current = head;
        while (current != null) {
            arlist.add(current.data);
            current = current.next;
        }

        int low = 0;
        int high = arlist.size() - 1;

        while (low < high) {
            if (!arlist.get(low).equals(arlist.get(high))) {
                return false; // Not a palindrome
            }
            low++;
            high--;
        }

        return true; // Is a palindrome
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        QuestionFour q1 = new QuestionFour();
        System.out.println("Enter the number of digits present in the
linked list:");
        int num = sc.nextInt();
        System.out.println("Enter the digits present in the linked
list:");
        for (int i = 0; i < num; i++) {
            int digit = sc.nextInt();
            q1.insertAtEnd(digit);
        }

        boolean isPalindrome = q1.isPalindrome();
        System.out.println(isPalindrome);

        sc.close();
    }
}
```

Output :-

```
Enter the number of digits present in the linked list:
4
Enter the digits present in the linked list:
1 2 2 1
true
```

Q5. . Given two numbers represented by two lists, write a function that returns the sum list. The sum list is a list representation of the addition of two input numbers.
Example:
Input: List1: 5->6->3// represents number 563
List2: 8->4->2// represents number 842
Output: Resultant list: 1->4->0->5// represents number 1405
Explanation: 563 + 842 = 1405
Input: List1: 7->5->9->4->6 // represents number 75946

List2: 8->4 // represents number 84
Output: Resultant list: 7->6->0->3->0//represents number 76030
Explanation: 75946+84-76030

Ans :-

```
package LinkedListAssignment;

import java.util.Scanner;

class LinkedList {
    Node head;

    class Node {
        int data;
        Node next;

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

    public void insertAtEnd(int newData) {
        Node newNode = new Node(newData);
        if (head == null) {
            head = newNode;
        } else {
            Node temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }
```

```java
public Node reverse(Node node) {
    Node prev = null;
    Node current = node;
    Node next = null;
    while (current != null) {
        next = current.next;
        current.next = prev;
        prev = current;
        current = next;
    }
    return prev;
}

public LinkedList addTwoLists(Node first, Node second) {
    first = reverse(first);
    second = reverse(second);

    LinkedList result = new LinkedList();
    int carry = 0;

    while (first != null || second != null || carry > 0) {
        int sum = carry;

        if (first != null) {
            sum += first.data;
            first = first.next;
        }

        if (second != null) {
            sum += second.data;
            second = second.next;
        }

        carry = sum / 10;
        int digit = sum % 10;

        Node newNode = new Node(digit);
```

```java
            newNode.next = result.head;
            result.head = newNode;
        }

        return result;
    }

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

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        LinkedList list1 = new LinkedList();
        LinkedList list2 = new LinkedList();

        System.out.println("Enter the number of digits for the first
linked list:");
        int n1 = sc.nextInt();
        System.out.println("Enter the digits for the first linked list:");
        for (int i = 0; i < n1; i++) {
            int digit = sc.nextInt();
            list1.insertAtEnd(digit);
        }

        System.out.println("Enter the number of digits for the second
linked list:");
        int n2 = sc.nextInt();
        System.out.println("Enter the digits for the second linked
list:");
        for (int i = 0; i < n2; i++) {
            int digit = sc.nextInt();
            list2.insertAtEnd(digit);
        }

        LinkedList result = new LinkedList();
        result = result.addTwoLists(list1.head, list2.head);

        System.out.println("Resultant list:");
        result.printList(result.head);

        sc.close();
    }
}
```

Output :-

```
Enter the number of digits for the first linked list:
3
Enter the digits for the first linked list:
5 6 3
Enter the number of digits for the second linked list:
3
Enter the digits for the second linked list:
8 4 2
Resultant list:
1 4 0 5
```