PROGRAM:

```
import java.util.*;
public class task4a {
    Queue<Integer> q1 = new LinkedList<>();
    Queue<Integer> q2 = new LinkedList<>();
     public void push(int x) {
        System.out.println("Pushing element: " + x);
        q2.add(x);
        while (!q1.isEmpty()) {
            q2.add(q1.poll());
        }
        swap();
    }
    private void swap() {
        Queue<Integer> temp = q1;
        q1 = q2;
        q2 = temp;
    }
   public void pop() {
        if (q1.isEmpty()) {
            System.out.println("Stack is empty. Cannot pop.");
            return;
        }
        System.out.println("Popping element: " + q1.poll());
    }
   public void top() {
        if (q1.isEmpty()) {
            System.out.println("Stack is empty. No top element.");
            return;
        }
        System.out.println("Top element: " + q1.peek());
```

```
    }
     public void isEmpty() {
        System.out.println("Is stack empty? " + q1.isEmpty());
    }
   public static void main(String[] args) {
        task4a stack = new task4a();
        stack.push(10);
        stack.push(20);
        stack.push(30);
        stack.push(40);
        stack.pop();
        stack.top();
        stack.isEmpty();
        System.out.println("Elements in the stack: " + stack.q1);
    }
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Varshinii\Documents\JAVASCRIPT\example1> java task4a.java
Pushing element: 10
Pushing element: 20
Pushing element: 30
Pushing element: 40
Popping element: 40
Top element: 30
Is stack empty? false
Elements in the stack: [30, 20, 10]
```

PROGRAM:

```
import java.util.ArrayList;

class BagOfNumbers {
    private ArrayList<Integer> bag = new ArrayList<>();

    public void add(int x) {
        bag.add(x);
        System.out.println("Added: " + x);
    }

    public void remove(int x) {
        if (bag.remove((Integer) x)) {
            System.out.println("Removing " + x + " from the bag...");
        } else {
            System.out.println("Number " + x + " not found in the bag.");
        }
    }

    public void countOccurrences(int x) {
        int count = 0;
        for (int num : bag) {
            if (num == x) count++;
        }
        System.out.println("Count occurrences of " + x + ": " + count);
    }

    public void isEmpty() {
        System.out.println("Is the bag empty? " + bag.isEmpty());
    }

    public void size() {
```

```
        System.out.println("Bag size: " + bag.size());
    }

    public void display() {
        System.out.println("Bag contents: " + bag);
    }

    public static void main(String[] args) {
        BagOfNumbers bag = new BagOfNumbers();

        System.out.println("Adding numbers: 5, 10, 5, 20");
        bag.add(5);
        bag.add(10);
        bag.add(5);
        bag.add(20);
        bag.display();

        bag.countOccurrences(5);
        bag.remove(5);
        bag.display();
        bag.size();
        bag.isEmpty();

        System.out.println("Removing all numbers...");
        bag.remove(10);
        bag.remove(5);
        bag.remove(20);
        bag.isEmpty();
    }
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Adding numbers: 5, 10, 5, 20
Added: 5
Added: 10
Added: 5
Added: 20
Bag contents: [5, 10, 5, 20]
Count occurrences of 5: 2
Removing 5 from the bag...
Bag contents: [10, 5, 20]
Bag size: 3
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Removing 5 from the bag...
Bag contents: [10, 5, 20]
Bag size: 3
Is the bag empty? false
Removing all numbers...
Removing 10 from the bag...
Removing 5 from the bag...
Removing 20 from the bag...
Is the bag empty? true
PS C:\Users\Varshinii\Documents\JAVASCRIPT\example1>
```

PROGRAM:

```
import java.util.*;

public class DiskTowerRecursion {
    static PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Collections.reverseOrder());
    static int currentMax;

    public static void solveTower(int[] disks, int day, int n) {
        if (day == n) return; // Base case: all days processed

        maxHeap.add(disks[day]);
        System.out.print("Day " + (day + 1) + ": ");

        while (!maxHeap.isEmpty() && maxHeap.peek() == currentMax) {
            System.out.print(maxHeap.poll() + " ");
            currentMax--;
        }
        System.out.println();

        solveTower(disks, day + 1, n);
    }

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

        System.out.print("Enter the number of disks (days): ");
        int n = sc.nextInt();
        int[] disks = new int[n];

        System.out.println("Enter the disk sizes:");
        for (int i = 0; i < n; i++) {
            disks[i] = sc.nextInt();
```

```
        }

        currentMax = Arrays.stream(disks).max().getAsInt();

        System.out.println("Tower construction order:");
        solveTower(disks, 0, n);
    }
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Enter the number of disks (days): 5
Enter the disk sizes:
4  5 1 2 3
Tower construction order:
Day 1:
Day 2: 5 4
Day 3:
Day 4:
Day 5: 3 2 1
```

PROGRAM:

```
class Insertion {


    static class Node {
        int data;
        Node next;

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

    Node head;


    public void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = newNode;
    }
```

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



public Node insertionSort(Node head) {

    Node dummy = new Node(0);
    Node curr = head;

    while (curr != null) {
        Node prev = dummy;
        Node next = curr.next;



        while (prev.next != null && prev.next.data < curr.data) {
            prev = prev.next;
        }

                curr.next = prev.next;
        prev.next = curr;



        curr = next;
    }

    return dummy.next;
}
```

```
    public static void main(String[] args) {
        Insertion list = new Insertion();



        list.insert(4);
        list.insert(2);
        list.insert(1);
        list.insert(3);

        System.out.print("Original List: ");
        list.printList(list.head);



        list.head = list.insertionSort(list.head);

        System.out.print("Sorted List: ");
        list.printList(list.head);
    }
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

        InsertionSortLinkedList list = new InsertionSortLinkedList();
                                            ^
  symbol:   class InsertionSortLinkedList
  location: class Insertion
2 errors
PS C:\Users\Varshinii\OneDrive\Documents\JAVASCRIPT\example1> javac Insertion.java
PS C:\Users\Varshinii\OneDrive\Documents\JAVASCRIPT\example1> java Insertion.java
Original List: 4 2 1 3
Sorted List: 1 2 3 4
PS C:\Users\Varshinii\OneDrive\Documents\JAVASCRIPT\example1>
```

PROGRAM:

```
class RemoveElementLinkedList {

    static class Node {
        int data;
        Node next;

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

    Node head;

    public void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = newNode;
    }

    public void printList() {
        Node temp = head;
        while (temp != null) {
```

```
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
public void remove(int key) {
        if (head == null) {
            return;
        }
      if (head.data == key) {
            head = head.next;
            return;
        }

        Node prev = null, curr = head;
        while (curr != null && curr.data != key) {
            prev = curr;
            curr = curr.next;
        }

        if (curr == null) {
            return;
        }
        prev.next = curr.next;
    }

    public static void main(String[] args) {
        RemoveElementLinkedList list = new RemoveElementLinkedList();
        list.insert(1);
        list.insert(2);
        list.insert(3);
        list.insert(4);
```

```
        list.insert(5);

        System.out.print("Original List: ");
        list.printList();



        list.remove(3);
        System.out.print("Updated List (after removing 3): ");
        list.printList();

        list.remove(1);
        System.out.print("Updated List (after removing 1): ");
        list.printList();



        list.remove(6);
        System.out.print("Updated List (after attempting to remove 6): ");
        list.printList();
    }
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Sorted List: 1 2 3 4
PS C:\Users\Varshinii\OneDrive\Documents\JAVASCRIPT\example1> javac RemoveElementLinkedList.ja
a
PS C:\Users\Varshinii\OneDrive\Documents\JAVASCRIPT\example1> java RemoveElementLinkedList.jav

Original List: 1 2 3 4 5
Updated List (after removing 3): 1 2 4 5
Updated List (after removing 1): 2 4 5
Updated List (after attempting to remove 6): 2 4 5
PS C:\Users\Varshinii\OneDrive\Documents\JAVASCRIPT\example1>
```

PROGRAM:

```
import java.util.HashSet;

class RemoveDuplicates {

    static class Node {
        int data;
        Node next;

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

    Node head;



    public void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = newNode;
    }
```

```
public void removeDuplicates() {
    HashSet<Integer> set = new HashSet<>();
    Node current = head;
    Node prev = null;

    while (current != null) {
        if (set.contains(current.data)) {

            prev.next = current.next;
        } else {
            set.add(current.data);
            prev = current;
        }
        current = current.next;
    }
}


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


public static void main(String[] args) {
    RemoveDuplicates list = new RemoveDuplicates();
```

```
        list.insert(1);
        list.insert(2);
        list.insert(2);
        list.insert(3);
        list.insert(4);
        list.insert(4);
        list.insert(5);

        System.out.print("Original List: ");
        list.display();

        list.removeDuplicates();

        System.out.print("List After Removing Duplicates: ");
        list.display();
    }
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Caused by: java.lang.ClassNotFoundException: RemoveDupllicates.java
PS C:\Users\Varshinii\OneDrive\Documents\JAVASCRIPT\example1> javac RemoveDupllicates.java
error: file not found: RemoveDupllicates.java
Usage: javac <options> <source files>
use --help for a list of possible options
PS C:\Users\Varshinii\OneDrive\Documents\JAVASCRIPT\example1> javac RemoveDuplicates.java
PS C:\Users\Varshinii\OneDrive\Documents\JAVASCRIPT\example1> java RemoveDuplicates.java
Original List: 1 2 2 3 4 4 5
List After Removing Duplicates: 1 2 3 4 5
PS C:\Users\Varshinii\OneDrive\Documents\JAVASCRIPT\example1>
```