Task 4

a)

import java.util.\*;

class StackUsingQueues {

Queue<Integer> q1 = new LinkedList<>();

Queue<Integer> q2 = new LinkedList<>();

// Push element onto stack

public void push(int x) {

// Step 1: Enqueue to q2

q2.add(x);

// Step 2: Move all elements from q1 to q2

while (!q1.isEmpty()) {

q2.add(q1.poll());

}

// Step 3: Swap q1 and q2

Queue<Integer> temp = q1;

q1 = q2;

q2 = temp;

}

// Pop element from stack

public int pop() {

if (q1.isEmpty()) return -1;

return q1.poll();

}

// Peek top element

public int top() {

if (q1.isEmpty()) return -1;

return q1.peek();

}

// Check if stack is empty

public boolean isEmpty() {

return q1.isEmpty();

}

public static void main(String[] args) {

StackUsingQueues stack = new StackUsingQueues();

stack.push(10);

stack.push(20);

stack.push(30);

System.out.println("Top: " + stack.top()); // 30

System.out.println("Pop: " + stack.pop()); // 30

System.out.println("Top: " + stack.top()); // 20

}

}

b)

import java.util.\*;

class BagOfNumbers {

private List<Integer> bag;

public BagOfNumbers() {

bag = new ArrayList<>();

}

// Add number to bag

public void add(int num) {

bag.add(num);

}

// Remove one occurrence of number

public boolean remove(int num) {

return bag.remove(Integer.valueOf(num));

}

// Check if bag contains number

public boolean contains(int num) {

return bag.contains(num);

}

// Count occurrences

public int count(int num) {

int cnt = 0;

for (int x : bag) {

if (x == num) cnt++;

}

return cnt;

}

// Show all numbers

public void display() {

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

}

public static void main(String[] args) {

BagOfNumbers bag = new BagOfNumbers();

bag.add(5);

bag.add(10);

bag.add(5);

bag.add(7);

bag.display(); // Bag: [5, 10, 5, 7]

System.out.println("Count of 5: " + bag.count(5)); // 2

bag.remove(10);

bag.display(); // Bag: [5, 5, 7]

}

}

c)

public class TowerOfHanoi {

// Recursive function to move n disks

public static void solveTower(int n, char from, char to, char aux) {

if (n == 1) {

System.out.println("Move disk 1 from " + from + " to " + to);

return;

}

// Move n-1 disks from source to auxiliary

solveTower(n - 1, from, aux, to);

// Move nth disk

System.out.println("Move disk " + n + " from " + from + " to " + to);

// Move n-1 disks from auxiliary to destination

solveTower(n - 1, aux, to, from);

}

public static void main(String[] args) {

int n = 3; // Number of disks

System.out.println("Tower of Hanoi with " + n + " disks:");

solveTower(n, 'A', 'C', 'B');

}

}