question 1

public void copyPaste(Queue src, int index) {

if (index < 0 || index >= N) {

return;

}

int srcCount = (src.last - src.first + src.N) % src.N;

int insertPosition = (first + index) % N;

int elementsToShift = (last - insertPosition + N) % N;

for (int i = elementsToShift - 1; i >= 0; i--) {

int targetIndex = (last + i + 1) % N;

int sourceIndex = (last + i) % N;

array[targetIndex] = array[sourceIndex];

}

for (int i = 0; i < srcCount; i++) {

int srcIndex = (src.first + i) % src.N;

int destIndex = (insertPosition + i) % N;

array[destIndex] = src.array[srcIndex];

}

last = (last + srcCount) % N;

}

question 2

public void Queue(Queue[] queues) {

this.first = null;

this.last = null;

for (Queue queue : queues) {

if (queue.first != null) {

Node currentSourceNode = queue.first;

while (currentSourceNode != null) {

Node newNode = new Node(currentSourceNode.data);

if (this.first == null) {

this.first = newNode;

} else {

this.last.next = newNode;

}

this.last = newNode;

currentSourceNode = currentSourceNode.next;

}

}

}

}

question 3

public void accumulateLeafNodes(Queue queue) {

if (this.root != null) {

accumulateLeafNodesRecursive(this.root, queue);

}

}

private void accumulateLeafNodes(Node node, Queue queue) {

if (node == null) {

return;

}

if (node.left == null && node.right == null) {

Element leafElement = new Element(node.data);

queue.enqueue(leafElement);

} else {

if (node.left != null) {

accumulateLeafNodesRecursive(node.left, queue);

}

if (node.right != null) {

accumulateLeafNodesRecursive(node.right, queue);

}

}

}