1- Consider the following statements: Array Queue queue = new Array Queue(); int x, y; Show what is output by the following segment of code: x = 4; y = 5; queue.enqueue(x); queue. enqueue(y); x = queue.front( ); queue.dequeue( ); queue. enqueue(x + 5); queue. enqueue(16); queue. enqueue(x); queue. enqueue(y - 3); system.out.println( "Queue Elements: "); while (!queue.isEmptyQueue()) { system.out.println(queue.front() ); queue.dequeue(); }

ArrayQueue<int> queue = new ArrayQueue();  
 int x, y;  
 x = 4;  
 y = 5;  
 queue.enqueue(x);  
 queue.enqueue(y);  
 x = queue.front();  
 queue.dequeue();  
 queue.enqueue(x + 5);  
 queue.enqueue(16);  
 queue.enqueue(x);  
 queue.enqueue(y - 3);  
 System.out.println("Queue Elements: ");  
 while (!queue.isEmptyQueue()) {  
 System.out.println(queue.front());  
 queue.dequeue();  
 }  
 Queue Elements:  
 9  
 16  
 4  
 2

2- What is the output of the following program segment? linkedQueue queue = new linkedQueue(); queue.enqueue(10); queue.enqueue(20); cout

LinkedQueue<Integer> queue = new LinkedQueue<>();  
 queue.enqueue(10);  
 queue.enqueue(20);  
 System.out.println(queue.front());  
 queue.dequeue();  
 queue.enqueue(2 \* queue.back());  
 queue.enqueue(queue.front());  
 queue.enqueue(5);  
 queue.enqueue(queue.back() - 2);  
 LinkedQueue<Integer> tempQueue = new LinkedQueue<>();  
 tempQueue = queue;  
 while (!tempQueue.isEmptyQueue()) {  
 System.out.println(tempQueue.front());  
 tempQueue.dequeue();  
 }  
 System.out.println(queue.front());  
 System.out.println(queue.back());  
 Output:  
 10  
 20  
 10  
 5  
 20

3- Consider the following statements: ArrayStack stack = new ArrayStack(); ArrayQueue queue = new ArrayQueue(); int x; Suppose the input is: 14 8 14 22 64 35 19 32 7 11 13 30 -999 Show what is written by the following segment of code: stack.push(0); queue.enqueue(0); system.out.println( x); while (x != -999) { switch (x % 4) { case 0: stack.push(x); break; case 1: if (!stack.isEmptyStack()) { system.out.println( "Stack Element = " ); system.out.println( stack.top()); stack.pop(); } else system.out.println( "Sorry, the stack is empty." ); break; case 2: queue.enqueue(x); break; case 3: if (!queue.isEmptyQueue()) { system.out.println( "Queue Element = " ); system.out.println( queue.front()); queue.dequeue(); } else system.out.println( "Sorry, the queue is empty." ); break; } //end switch system.out.println( x); } //end while system.out.println( "Stack Elements: "); while (!stack.isEmptyStack()) { system.out.println( stack.top() ); stack.pop(); } system.out.println( "Queue Elements: "); while (!queue.isEmptyQueue()) {system.out.println( queue.front() ); queue.dequeue(); }

ArrayStack<int> stack = new ArrayStack();  
 ArrayQueue<int> queue = new ArrayQueue();  
 int x;  
*// Suppose the input is: 14 8 14 22 64 35 19 32 7 11 13 30 -999* stack.push(0);  
 queue.enqueue(0);  
 System.out.println(x);  
 while (x != -999) {  
 switch (x % 4) {  
 case 0:  
 stack.push(x);  
 break;  
 case 1:  
 if (!stack.isEmptyStack()) {  
 System.out.println("Stack Element = ");  
 System.out.println(stack.top());  
 stack.pop();  
 } else {  
 System.out.println("Sorry, the stack is empty.");  
 }  
 break;  
 case 2:  
 queue.enqueue(x);  
 break;  
 case 3:  
 if (!queue.isEmptyQueue()) {  
 System.out.println("Queue Element = ");  
 System.out.println(queue.front());  
 queue.dequeue();  
 } else {  
 System.out.println("Sorry, the queue is empty.");  
 }  
 break;  
 }  
 System.out.println(x);  
 }  
 System.out.println("Stack Elements: ");  
 while (!stack.isEmptyStack()) {  
 System.out.println(stack.top());  
 stack.pop();  
 }  
 System.out.println("Queue Elements: ");  
 while (!queue.isEmptyQueue()) {  
 System.out.println(queue.front());  
 queue.dequeue();  
 }

4- Suppose that queue is a queueType object and the size of the array implementing queue is 100. Also, suppose that the value of queueFront is 50 and the value of queueRear is 99. a- What are the values of queueFront and queueRear after adding an element to queue?

class QueueType {  
 تعريف المتغيرات المطلوبة *//* private int[] queueArray;  
 private int maxSize;  
 private int queueFront;  
 private int queueRear;  
  
 البناء األساسي للصف *//* public QueueType(int size) {  
 maxSize = size;  
 queueArray = new int[maxSize];  
 queueFront = 50;  
 queueRear = 99;  
 }  
  
 دالة إلضافة عنصر إلى الصف *//* public void enqueue(int item) {  
 if (queueRear == maxSize - 1) {  
 إلى نهاية المصفوفة، يعود للبداي queueRear عندما يصل *// ;0 = queueRear* ة  
 } else {  
 queueRear++;  
 }  
 queueArray[queueRear] = item;  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
 QueueType queue = new QueueType(100);  
  
 إضافة عنصر إلى الصف *//* queue.enqueue(10);  
  
 بعد إضافة العنصر queueRear و queueFront طباعة القيم الجديدة لـ *//* System.*out*.println("queueFront: " + queue.queueFront);  
 System.*out*.println("queueRear: " + queue.queueRear);  
 }  
}

b- What are the values of queueFront and queueRear after removing an element from queue?

class QueueType {  
 تعريف المتغيرات المطلوبة *//* private int[] queueArray;  
 private int maxSize;  
 private int queueFront;  
 private int queueRear;  
  
 البناء األساسي للصف *//* public QueueType(int size) {  
 maxSize = size;  
 queueArray = new int[maxSize];  
 queueFront = 50;  
 queueRear = 99;  
 }  
  
 دالة إلضافة عنصر إلى الصف *//* public void enqueue(int item) {  
 if (queueRear == maxSize - 1) {  
 إلى نهاية المصفوفة، يعود للبداي queueRear عندما يصل *// ;0 = queueRear* ة  
 } else {  
 queueRear++;  
 }  
 queueArray[queueRear] = item;  
 }  
  
 دالة إلزالة عنصر من الصف *//* public int dequeue() {  
 int removedItem = queueArray[queueFront];  
 if (queueFront == maxSize - 1) {  
 إلى نهاية المصفوفة، يعود للبد queueFront عندما يصل *// ;0 = queueFront* اية  
 } else {  
 queueFront++;  
 }  
 return removedItem;  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
 QueueType queue = new QueueType(100);  
  
 إزالة عنصر من الصف *//* int removedItem = queue.dequeue();  
  
 بعد إزالة العنصر queueRear و queueFront طباعة القيم الجديدة لـ *//* System.*out*.println("queueFront: " + queue.queueFront);  
 System.*out*.println("queueRear: " + queue.queueRear);  
 }  
}

5- Suppose that queue is a queueType object and the size of the array implementing queue is 100. Also, suppose that the value of queueFront is 99 and the value of queueRear is 25. a- What are the values of queueFront and queueRear after adding an element to queue? b- What are the values of queueFront and queueRear after removing an element from queue?

class QueueType {  
 تعريف المتغيرات المطلوبة *//* private int[] queueArray;  
 private int maxSize;  
 private int queueFront;  
 private int queueRear;  
  
 البناء األساسي للصف *//* public QueueType(int size) {  
 maxSize = size;  
 queueArray = new int[maxSize];  
 queueFront = 99;  
 queueRear = 25;  
 }  
  
 دالة إلضافة عنصر إلى الصف *//* public void enqueue(int item) {  
 if (queueRear == maxSize - 1) {  
 إلى نهاية المصفوفة، يعود للبداي queueRear عندما يصل *// ;0 = queueRear* ة  
 } else {  
 queueRear++;  
 }  
 queueArray[queueRear] = item;  
 }  
  
 دالة إلزالة عنصر من الصف *//* public int dequeue() {  
 int removedItem = queueArray[queueFront];  
 if (queueFront == maxSize - 1) {  
 إلى نهاية المصفوفة، يعود للبد queueFront عندما يصل *// ;0 = queueFront* اية  
 } else {  
 queueFront++;  
 }  
 return removedItem;  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
 QueueType queue = new QueueType(100);  
  
 إضافة عنصر إلى الصف *//* queue.enqueue(10);  
  
 بعد إضافة العنصر queueRear و queueFront طباعة القيم الجديدة لـ *//* System.*out*.println("queueFront: " + queue.queueFront);  
 System.*out*.println("queueRear: " + queue.queueRear);  
  
 إزالة عنصر من الصف *//* int removedItem = queue.dequeue();  
  
 بعد إزالة العنصر queueRear و queueFront طباعة القيم الجديدة لـ *//* System.*out*.println("queueFront: " + queue.queueFront);  
 System.*out*.println("queueRear: " + queue.queueRear);  
 }  
}

6- Suppose that queue is a queueType object and the size of the array implementing queue is 100. Also, suppose that the value of queueFront is 25 and the value of queueRear is 75. a- What are the values of queueFront and queueRear after adding an element to queue? b- What are the values of queueFront and queueRear after removing an element from queue

class QueueType {  
 private int[] queueArray;  
 private int maxSize;  
 private int queueFront;  
 private int queueRear;  
  
 public QueueType(int size) {  
 maxSize = size;  
 queueArray = new int[maxSize];  
 queueFront = 25;  
 queueRear = 75;  
 }  
  
 public void enqueue(int item) {  
 if (queueRear == maxSize - 1) {  
 queueRear = 0;  
 } else {  
 queueRear++;  
 }  
 queueArray[queueRear] = item;  
 }  
  
 public int dequeue() {  
 int removedItem = queueArray[queueFront];  
 if (queueFront == maxSize - 1) {  
 queueFront = 0;  
 } else {  
 queueFront++;  
 }  
 return removedItem;  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
 QueueType queue = new QueueType(100);  
  
 *// Adding an element to the queue* queue.enqueue(10);  
  
 *// Printing the new values of queueFront and queueRear after adding the eleme* nt  
 System.*out*.println("queueFront: " + queue.queueFront);  
 System.*out*.println("queueRear: " + queue.queueRear);  
  
 *// Removing an element from the queue* int removedItem = queue.dequeue();  
  
 *// Printing the new values of queueFront and queueRear after removing the ele* ment  
 System.*out*.println("queueFront: " + queue.queueFront);  
 System.*out*.println("queueRear: " + queue.queueRear);  
 }  
}

7- Suppose that queue is a queueType object and the size of the array implementing queue is 100. Also, suppose that the value of queueFront is 99 and the value of queueRear is 99. a- What are the values of queueFront and queueRear after adding an element to queue? b- What are the values of queueFront and queueRear after removing an element from queue?

class QueueType {  
 private int[] queueArray;  
 private int maxSize;  
 private int queueFront;  
 private int queueRear;  
  
 public QueueType(int size) {  
 maxSize = size;  
 queueArray = new int[maxSize];  
 queueFront = 99;  
 queueRear = 99;  
 }  
  
 public void enqueue(int item) {  
 if (queueRear == maxSize - 1) {  
 queueRear = 0;  
 } else {  
 queueRear++;  
 }  
 queueArray[queueRear] = item;  
 }  
  
 public int dequeue() {  
 int removedItem = queueArray[queueFront];  
 if (queueFront == maxSize - 1) {  
 queueFront = 0;  
 } else {  
 queueFront++;  
 }  
 return removedItem;  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
 QueueType queue = new QueueType(100);  
  
 *// Adding an element to the queue* queue.enqueue(10);  
  
 *// Printing the new values of queueFront and queueRear after adding the eleme* nt  
 System.*out*.println("queueFront: " + queue.queueFront);  
 System.*out*.println("queueRear: " + queue.queueRear);  
  
 *// Removing an element from the queue* int removedItem = queue.dequeue();  
  
 *// Printing the new values of queueFront and queueRear after removing the ele* ment  
 System.*out*.println("queueFront: " + queue.queueFront);  
 System.*out*.println("queueRear: " + queue.queueRear);  
 }  
}

8- Write a function, reverseQueue, that takes as a parameter a queue object and uses a stack object to reverse the elements of the queue.

import java.util.Queue;  
import java.util.Stack;  
public class Main {  
 public static void reverseQueue(Queue<Integer> queue) {  
 Stack<Integer> stack = new Stack<>();  
  
 *// Push all elements from the queue to the stack* while (!queue.isEmpty()) {  
 stack.push(queue.remove());  
 }  
  
 *// Pop all elements from the stack and add them back to the queue* while (!stack.isEmpty()) {  
 queue.add(stack.pop());  
 }  
 }  
  
 public static void main(String[] args) {  
 *// Create a sample queue* Queue<Integer> queue = new LinkedList<>();  
 queue.add(1);  
 queue.add(2);  
 queue.add(3);  
 queue.add(4);  
 queue.add(5);  
  
 *// Print the original queue* System.*out*.println("Original Queue: " + queue);  
  
 *// Reverse the queue  
 reverseQueue*(queue);  
  
 *// Print the reversed queue* System.*out*.println("Reversed Queue: " + queue);  
 }  
}

9- Suppose an initially empty queue Q has performed a total of 32 enqueue operations, 10 first operations, and 15 dequeue operations, 5 of which returned null to indicate an empty queue. What is the current size of Q? Based on the information provided, we can calculate the current size of queue Q as follows: • Total enqueue operations: 32 • Total dequeue operations: 15 • Dequeue operations that returned null: 5 To calculate the current size of the queue, we subtract the total dequeue operations from the total enqueue operations and then subtract the number of dequeue operations that returned null. Current size of Q = Total enqueue operations - Total dequeue operations - Null dequeue operations Current size of Q = 32 - 15 - 5 Current size of Q = 12 Therefore, the current size of queue Q is 12. If you have any more questions, feel free to ask!

public class Main {  
 public static void main(String[] args) {  
 Deque<Integer> deque = new LinkedList<>();  
 deque.addFirst(3);  
 deque.addLast(8);  
 deque.addLast(9);  
 deque.addFirst(1);  
 System.*out*.println("last(): " + deque.getLast());  
 System.*out*.println("isEmpty(): " + deque.isEmpty());  
 deque.addFirst(2);  
 System.*out*.println("removeLast(): " + deque.removeLast());  
 deque.addLast(7);  
 System.*out*.println("first(): " + deque.getFirst());  
 System.*out*.println("last(): " + deque.getLast());  
 deque.addLast(4);  
 System.*out*.println("size(): " + deque.size());  
 System.*out*.println("removeFirst(): " + deque.removeFirst());  
 System.*out*.println("removeFirst(): " + deque.removeFirst());  
 }  
}