Section: 1

Q 1 = a

Q 2 = d

Q 3 = d

Q 5 = c

Q 6 = c

Q 7 = c

Q 8 = d

Q 9 = c

Q10 = d

Section: 2

Q 2

import java.util.Scanner;

/\*\*

\* Q2: Write a JAVA code to implement a queue with following two operations :-

\* enQueue() :- Adds an element at the back of the queue. deQueue() :- Deletes

\* the element at the front of the queue

\*

\* @author Shivam

\*

\*/

class Queue {

int start;

int end;

int size;

int capacity;

int array[];

public Queue(int capacity) {

this.capacity = capacity;

start = this.size = 0;

end = capacity - 1;

array = new int[this.capacity];

}

boolean isEmpty(Queue queue) {

return (queue.size == 0);

}

void enQueue(int data) {

this.end = (this.end + 1) % this.capacity;

this.array[this.end] = data;

this.size = this.size + 1;

}

void deQueue() {

if (isEmpty(this))

return;

this.start = (this.start + 1) % this.capacity;

this.size = this.size - 1;

}

void print() {

for (int i = start; i <= end; i++) {

System.out.print(array[i] + " ");

}

}

}

public class Mainline {

public static void main(String[] args) {

Queue queue = new Queue(100);

Scanner s = new Scanner(System.in);

int noe = s.nextInt();

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

queue.enQueue(s.nextInt());

}

int deo = s.nextInt();

if (deo > queue.size) {

System.out.println("Sorry no operation can be performed");

} else {

while (deo-- > 0) {

queue.deQueue();

deo--;

}

}

queue.print();

s.close();

}

}