Question 1

[20 marks]

1 Write a Java program given the following specification and provide comments which explain how your algorithm works.

Problem Statement
The goal is to read in a list of integers into an array and output the one which occurs most frequently. If there are two or more values that occur most frequently then choose the one which is lower.

Input Format
The first line contains *N*, the number of inputs. The second line contains *N* integers, $a_0, a_1,...a_{N-1}$, separated by a space.

Output Format
The mode, that is, the number which occurs most frequently in the list.

Constraints
$1 \leq N \leq 1000$
$-1000 \leq a_i \leq 1000$

Sample Input
7
15 18 3 2 -5 6 2 6

Sample Output
2

```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Map;
import java.util.HashMap;

public class Q1 {
    public static void main (String args[]) {
        ArrayList<Integer> numArray = new ArrayList<Integer>();
        Scanner sc = new Scanner(System.in);

        // the number of inputs
        int N = sc.nextInt();
        for (int i = 0; i < N; i++) {
            int inputNum = sc.nextInt();
            numArray.add(inputNum);
        }
```

```
        sc.close();

        //Use HashMap to record the repeat times of each INT
        // <Integer Value, Repeat Times>
        Map<Integer, Integer> frequencyMap = new HashMap<Integer, Integer>();

        // Iterate the ArrayList
       for (int number : numArray) {
        // and calculate the repeat times
          frequencyMap.put(number, frequencyMap.getOrDefault(number, 0) + 1);
       }

       int smallerNumber = Integer.MAX_VALUE;
       int maxFrequency = 0;

       // Convert Map to KEY-VALUE Entry
       for (Map.Entry<Integer, Integer> entry : frequencyMap.entrySet()) {
          int currentNumber = entry.getKey();
          int currentFrequency = entry.getValue();

          // If the repeat frequency is higher
          // Or same frequency and smaller number
          if (currentFrequency > maxFrequency
                || (currentFrequency == maxFrequency
                    && currentNumber < smallerNumber))
          {
              smallerNumber = currentNumber;
              maxFrequency = currentFrequency;
           }
       }

    // Print out the smaller number occurs most frequently
       System.out.println(smallerNumber);
    }
}
```

CS210-2016-Autumn

Question 2

[20 marks]

2 Write a Java program given the following specification and provide comments which explain how your algorithm works.

Problem Statement
Use a **priority queue** to sort a sequence of numbers. You must write your own Priority Queue class. The numbers are inserted into the priority queue, which assigns higher numbers higher priority. The numbers are then removed in a sorted order.

Input Format
The first line contains N, the number of inputs. The second line contains N integers, $a_0, a_1, ...a_{N-1}$, separated by a space.

Output Format
A line of sorted integers, from higher to lower, with each integer separated by a space.

Constraints
$1 \leq N \leq 1000$
$-1000 \leq a_i \leq 1000$

Sample Input
5
8 2 7 5 9

Sample Output
9 8 7 5 2

Answer 1 – Interface

```
import java.util.Scanner;
import java.util.Comparator;
import java.util.Queue;
import java.util.PriorityQueue;

public class Q2_Interface {
   public static void main (String args[]) {
      Scanner sc = new Scanner(System.in);
      Queue<Integer> pq = new PriorityQueue<Integer>(
         new Comparator<Integer>() {
            @Override
            public int compare(Integer o1, Integer o2) {
               return o2 - o1;
         }
      });
```

```
            // Number of Inputs
        int N = Integer.parseInt(sc.nextLine());
        // N integers separated by a space.
        for(int i = 0; i < N ; i++) {
            int inputNum = sc.nextInt();
            pq.add(inputNum);
        }
        sc.close();

        // Print out sorted Integer
        while(!pq.isEmpty()) {
            System.out.print(pq.remove() + " ");
        }
    }
}
```

Answer 2 – Full Queue Class

```
import java.util.Scanner;

public class Q2_FullPriorityQueueClass {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        FullPriorityQueue pq = new FullPriorityQueue(100);

        // Number of Inputs
        int N = Integer.parseInt(sc.nextLine());

        // N integers separated by a space.
        for(int i = 0; i < N ; i++) {
            int inputNum = sc.nextInt();
            pq.insert(inputNum);
        }
        sc.close();

        // Print out sorted Integer
        while(!pq.isEmpty()) {
            System.out.print(pq.remove() + " ");
        }
    }
}
```

CS210-2016-Autumn

```
class FullPriorityQueue{
    private int maxSize;
    private int[] queArray;
    private int front;
    private int nItems;

    public FullPriorityQueue(int s) { // constructor
        maxSize = s;
        queArray = new int[maxSize];
        front = 0;
        nItems = 0;
    }

    // insert item from large to small
    public boolean insert(int item) {
        if (nItems == 0) {
            // If queue is empty, insert it directly.
            queArray[nItems++] = item;
        } else {
            int position; // the position to insert
            // find the postion from back to front
            for (position = nItems - 1; position >= 0; position--) {
                // if element is smaller, move element backwards
                if (item > queArray[position]) {
                    queArray[position + 1] = queArray[position];
                // find the element, end the loop.
                } else {
                    break;
                }
            }
            // insert the element in that slot.
            queArray[position + 1] = item;
            nItems++;
        }
        return true; //successfully inserted
    }


    public boolean isEmpty() { // true if queue is empty
        return (nItems==0);
    }
```

CS210-2016-Autumn

```
    public int remove() { // take item from front of queue
        if(isEmpty()) return (Integer) null; //don't remove if empty
        int temp = queArray[front];// get value and incr front
        front++;
        // deal with wraparound
        if(front == maxSize) front = 0;
        nItems--; // one less item
        return temp;
    }
}
```

*Question 3

[20 marks]

3 Write a Java method that takes in a Linked List object (double-ended and doubly-linked) and deletes the first half of the linked list. If there is an odd number of links then the middle link should also be deleted. The method then returns the Linked List object. Provide comments which explain how your algorithm works.

```
class Node {
    int data;
    Node next;
    Node prev;

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

CS210-2016-Autumn

```java
class DoublyLinkedList {
    Node head;
    Node tail;

    // Method to delete the first half of the doubly-linked list
    public DoublyLinkedList deleteFirstHalf() {
        if (head == null) {
            // If the list is empty, nothing to delete
            return this;
        }
        // Count the number of nodes in the list
        int count = 0;
        Node current = head;
        while (current != null) {
            count++;
            current = current.next;
        }

        // Find the middle node (if the count is odd,
        middle node will be exactly at the center)
        int middle = 0;
        if(count % 2 == 1) {
            //If nodes number is odd, middle node should also be deleted
            middle = count / 2 + 1;
        }
        else {
            middle = count / 2;
        }

        // Delete nodes until reaching the middle
        current = head;
        for (int i = 0; i < middle; i++) {
            current = current.next;
        }

        // Adjust pointers to delete the first half
        current.prev.next = null;
        current.prev = null;
        head = current;

        return this;
    }
}
```

CS210-2016-Autumn

Question 4

Question a

[20 marks]

4 a) Identify the output that the following Java code produces and explain your reasoning clearly. [10 marks]

```
public class Recursion{
    public static void main(String[] args){
        System.out.println(method(3));
    }

    public static int method(int number){
        if (number > 30){
            return 7;
        }
        System.out.println("hello");
        return method(number + 7) - 8;
    }
}
```

The program runs main function first, it will call `method(3)`

1) method(3).

3 < 30 => skip if statement

Print "hello", then change line

return method(3 + 7) – 8 = method(10) - 8

2) method(10).

10 < 30 => skip if statement

Print "hello" , then change line

return method(10 + 7) – 8 = method(17) - 8

3) method(17).

17 < 30 => skip if statement

Print "hello" , then change line

return method(17 + 7) – 8 = method(24) - 8

4) method(24).

24 < 30 => skip if statement

Print "hello" , then change line

return method(24 + 7) – 8 = method(31) - 8

CS210-2016-Autumn

5) method(31).
31 > 30 => run if statement
return 7
⇨ method(31) = 7

6) Calling method(24)
method(24) = method(31) – 8 = 7 – 8 = -1

7) Calling method(17)
method(17) = method(17) – 8 = -1 – 8 = -9

8) Calling method(10)
method(10) = method(10) – 8 = -9 – 8 = -17

9) Calling method(3)
method(3) = method(3) – 8 = -17 – 8 = -25

Therefore, the Java Program outputs

```
hello
hello
hello
hello
-25
```

when it runs.

CS210-2016-Autumn

Question b

b) Identify the output that the following Java code produces and explain your reasoning clearly. [10 marks]

```
public class BitManipulation{
    public static void main (String[] args){
        System.out.println((7&11)<<7);

    }
}
```

The program will print out the equation

(7&11)<<7)

Step 1: 7 & 11

$(7)_{10}$	$=(00000111)_2$	
$(11)_{10}$	$=(00001011)_2$	&
	$`(00000011)_2$	$= (3)_{10}$

Step 2: 3 << 7

$(0000000011)_2$ << 6 = $(0110000000)_2$ = $(384)_{10}$

Therefore, the Java Program outputs 384 when it runs.