Question 1

[25 marks]

1 **Problem Statement** [25 marks]

Write a Java program that reads in a list of numbers, and sorts them according to the number of steps they follow in the Collatz sequence before reaching 1 (most steps comes first). A Collatz sequence starts with a given number and follows the operation below until reaching 1:

- If the number is even, divide it by two.
- If the number is odd, triple it and add one.

State the **Big-O complexity** of the algorithm you have written, and explain what this means in your own words.

Sample Input
6
2
8
13
15

Sample Output
15
13
6
8
2

Explanation
15 ...(takes 17 steps to reach 1)
13 ...(takes 9 steps to reach 1)
6 ...(takes 8 steps to reach 1)
8 ...(takes 3 steps to reach 1)
2 ...(takes 1 step to reach 1)

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

public class Q1 {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
```

CS210-2019-January

```
        Queue<Integer> pq = new PriorityQueue<Integer>(
            new Comparator<Integer>() {
            @Override
            public int compare(Integer o1, Integer o2) {
                return getCollatzSteps(o2) - getCollatzSteps(o1);
            }
        });

        while(true) {
            String inputLine = sc.nextLine();
            if(inputLine.isEmpty()) {
                sc.close();
                break;
            }
            int inputNum = Integer.parseInt(inputLine);
            pq.add(inputNum); // O(n*logn) n: number of input
        }

        while (!pq.isEmpty()) {
            System.out.println(pq.poll());
        }
    }

    public static int getCollatzSteps (int input) {
        int steps = 0;
        while(input>1) { //
            if(input % 2 == 0) {
                input = input / 2;
            }
            else {
                input = input * 3 + 1;
            }
            steps++;
        }
        return steps;
    }
}
/*
   Big-O Complexity is time complexity,  it is a concept that describes
    the speed of an algorithm according to the size of input.
   The Big-O Complexity in my program is O(n*logn)
 */
```

CS210-2019-January

Question 2

[25 marks]

2 Problem Statement [25 marks]

Four horses are running a race, with the following probabilities of winning the race:

Horse A: 53%
Horse B: 26%
Horse C: 14%
Horse D: 7%

Write a Monte Carlo simulation which estimates the probability that Horse B will finish third.

```
public class Q2 {

    public static void main(String[] args) {
        int totalSimulations = 1000000; // Number of simulations
        int countBThirdPlace = 0; // Counter for Horse B finishing third

        for (int i = 0; i < totalSimulations; i++) { // A[0,53) B[53,79) C[79,93) D 93,100]
            double randomValue = Math.random() * 100; // Generate a random number between 0 and 100

            // Determine First Place Horse
            char firstPlace = (randomValue < 53) ? 'A' : (randomValue < 79) ? 'B' :
(randomValue < 93) ? 'C' : 'D';

            // Determine Second Place Horse (exclude First Place)
            randomValue = Math.random() * (100 - getProbability(firstPlace));
            char secondPlace;
            if (firstPlace == 'A') { // B[0,26) C[26,40) D[40,47]
                secondPlace = (randomValue < 26) ? 'B' : (randomValue < 40) ? 'C' : 'D';
            } else if (firstPlace == 'B') { // A[0,53) C[53,67) D[67,74]
                secondPlace = (randomValue < 53) ? 'A' : (randomValue < 67) ? 'C' : 'D';
            } else if (firstPlace == 'C') { // A[0,53) B[53,79) D[79,86]
                secondPlace = (randomValue < 53) ? 'A' : (randomValue < 79) ? 'B' : 'D';
            } else { // A[0,53) B[53,79) C[79,93]
                secondPlace = (randomValue < 53) ? 'A' : (randomValue < 79) ? 'B' : 'C';
            }

            // Determine Third Place Horse (exclude First Place and Second Place)
```

```
            randomValue = Math.random() * (100 - getProbability(firstPlace) -
getProbability(secondPlace));
            char thirdPlace;
            if ((firstPlace == 'A' && secondPlace == 'B') || (firstPlace == 'B' &&
secondPlace == 'A')) {
                thirdPlace = (randomValue < 14) ? 'C' : 'D'; // C[0,14) D[14,21]
            } else if ((firstPlace == 'A' && secondPlace == 'C') || (firstPlace == 'C' &&
secondPlace == 'A')) {
                thirdPlace = (randomValue < 26) ? 'B' : 'D'; // B[0,26) D[26,33]
            } else if ((firstPlace == 'A' && secondPlace == 'D') || (firstPlace == 'D' &&
secondPlace == 'A')) {
                thirdPlace = (randomValue < 26) ? 'B' : 'C'; // B[0,26) C[26,40]
            } else if ((firstPlace == 'B' && secondPlace == 'C') || (firstPlace == 'C' &&
secondPlace == 'B')) {
                thirdPlace = (randomValue < 53) ? 'A' : 'D'; // A[0,53) D[53,60]
            } else if ((firstPlace == 'B' && secondPlace == 'D') || (firstPlace == 'D' &&
secondPlace == 'B')) {
                thirdPlace = (randomValue < 53) ? 'A' : 'C'; // A[0,53) C[53,67]
            }else {
                thirdPlace = (randomValue < 53) ? 'A' : 'B'; // A[0,53) B[53,79]
            }
            // If Horse B finishes third, increment the counter
            if (thirdPlace == 'B') {
                countBThirdPlace++;
            }
        }

        // Calculate and print the estimated probability
        double estimatedProbability = (double) 100 * countBThirdPlace / totalSimulations;
        System.out.printf("Estimated probability that Horse B finishes third: %.2f %%",
        estimatedProbability);
    }

    // Helper method to get probability based on horse
    private static double getProbability(char horse) {
        switch (horse) {
            case 'A': return 53;
            case 'B': return 26;
            case 'C': return 14;
            case 'D': return 7;
            default:  return 0;
        }
    }
}
```

CS210-2019-January

Question 3

[25 marks]

3 **Problem Statement** [25 marks]

Manipulate a queue according to the given insert and remove commands and then output the number that is at the front of the queue. If a remove command is issued for an empty queue then nothing should happen. In your answer you should provide the full queue class.

Input Format
A series of lines involving either INSERT or REMOVE commands. The command INSERT is followed by a space and then a number to insert (e.g. INSERT 56).

Output Format
Output the number that is at the front of the queue following the given commands. If the queue is empty then output 0.

Sample Input
INSERT 56
INSERT 33
REMOVE
INSERT 83
REMOVE

Sample Output
83

Answer 1 – Interface

```
import java.util.Queue;
import java.util.LinkedList;
import java.util.Scanner;

public class Q3_Interface {
    public static void main (String args[]) {
        Queue<Integer> q = new LinkedList<Integer>();
        Scanner sc = new Scanner(System.in);

        while(true) {
            String input = sc.nextLine();
            if(input.isEmpty()) {
                sc.close();
                break;
            }
```

CS210-2019-January

```
            if(input.split(" ")[0].toUpperCase().equals("INSERT")){
                String insertNumStr = input.split(" ")[1];
                int insertNum = Integer.parseInt(insertNumStr);
                q.add(insertNum);
            }

            if(input.toUpperCase().equals("REMOVE")) {
            /* If a remove command is issued for an empty queue
               then nothing should happen. */
                if(!q.isEmpty()) q.remove();
            }
        }
        System.out.println(q.peek());
    }
}
```

Answer 2 – Full Queue Class

```
import java.util.Scanner;

public class Q3_FullQueueClass {
    public static void main (String args[]) {
        FullQueue q = new FullQueue(100);
        Scanner sc = new Scanner(System.in);

        while(true) {
            String input = sc.nextLine();
            if(input.isEmpty()) {
                sc.close();
                break;
            }
            if(input.split(" ")[0].toUpperCase().equals("INSERT")){
                String insertNumStr = input.split(" ")[1];
                int insertNum = Integer.parseInt(insertNumStr);
                q.insert(insertNum);
            }
            if(input.toUpperCase().equals("REMOVE")) {
            /* If a remove command is issued for an empty queue
               then nothing should happen. */
                if(!q.isEmpty()) q.remove();
            }
        }  // End of while statement
```

```
        System.out.println(q.remove());
    }
}

class FullQueue{
    private int maxSize;
    private long[] queArray;
    private int front;
    private int rear;
    private int nItems;

    public FullQueue(int s) { // constructor
        maxSize = s;
        queArray = new long[maxSize];
        front = 0;
        rear = -1;
        nItems = 0;
    }

  public boolean insert(long j) { // put item at rear of queue
        if(isFull()) return false; //don't remove if full
        // deal with wraparound
        if(rear == maxSize - 1) {
            rear = -1;// deal with wraparound
        }
        rear++;
        queArray[rear] = j; // increment rear and insert
        nItems++; // one more item
        return true; //successfully inserted
    }

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

    public long peekFront(){ // peek at front of queue
        return queArray[front];
    }
```

CS210-2019-January

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

    public boolean isFull() { // true if queue is full
        return (nItems==maxSize);
    }

    public int size() { // number of items in queue
        return nItems;
    }
}
```

CS210-2019-January

Question 4

Question a

[25 marks]

4 (a) Identify the output that the following Java code produces and explain your reasoning clearly. [6 marks]

```
public class Recursion{

   public static void main(String[] args){
      System.out.println(function("Launch"));
   }

   public static String function(String input){
      System.out.println("Evaluating");
      if(input.length()%7==0){
         return "Exit";
      }
      return(function(input+"Return")+"Terminated");
   }
}
```

The program runs main function first, it will call `function("Launch")`

1) function("Launch").

print out **"Evaluating", then change line.**

length = 6, 6 % 7 != 0 => skip if statement

return(function("LaunchReturn")+"Terminated")

2) function("LaunchReturn").

print out **"Evaluating", then change line.**

length = 12, 12 % 7 != 0 => skip if statement

return(function("LaunchReturnReturn")+"Terminated")

3) function("LaunchReturnReturn").

print out **"Evaluating", then change line.**

length = 18, 18 % 7 != 0 => skip if statement

return(function("LaunchReturnReturnReturn")+"Terminated")

4) function("LaunchReturnReturnReturn").

print out **"Evaluating", then change line.**

length = 24, 24 % 7 != 0 => skip if statement

return(function("LaunchReturnReturnReturnReturn")+"Terminated")

5) function("LaunchReturnReturnReturnReturn").

print out **"Evaluating", then change line.**

length = 30, 30 % 7 != 0 => skip if statement

CS210-2019-January

return(function(“LaunchReturnReturnReturnReturnReturn”)+"Terminated")

6) function(“LaunchReturnReturnReturnReturnReturn”).
print out “Evaluating”, then change line.
length = 36, 36 % 7 != 0 => skip if statement
return(function(“LaunchReturnReturnReturnReturnReturnReturn”)+"Terminated")

7) function(“LaunchReturnReturnReturnReturnReturnReturn”).
print out “Evaluating”, then change line.
length = 42, 42 % 7 == 0 => run if statement
return(“Exit”)

8) Calling function(“LaunchReturnReturnReturnReturnReturnReturn”)+"Terminated"
Get “ExitTerminated”

9) Calling function(“LaunchReturnReturnReturnReturnReturn”)+"Terminated"
Get “ExitTerminatedTerminated”

10) Calling function(“LaunchReturnReturnReturnReturn”)+"Terminated"
Get “ExitTerminatedTerminatedTerminated”

11) Calling function(“LaunchReturnReturnReturn”)+"Terminated"
Get “ExitTerminatedTerminatedTerminatedTerminated”

12) Calling function(“LaunchReturnReturn”)+"Terminated"
Get “ExitTerminatedTerminatedTerminatedTerminatedTerminated”

13) Calling function(“LaunchReturn”)+"Terminated"
Get “ExitTerminatedTerminatedTerminatedTerminatedTerminatedTerminated”
Finally, print out “ExitTerminatedTerminatedTerminatedTerminatedTerminatedTerminated”, then change line.

Therefore, the Java Program outputs

```
Evaluating
Evaluating
Evaluating
Evaluating
Evaluating
Evaluating
Evaluating
ExitTerminatedTerminatedTerminatedTerminatedTerminatedTerminated
```

when it runs.

CS210-2019-January

Question b

(b) Identify the output that the following Java code produces and explain your reasoning clearly. [6 marks]

```
public class BitManipulation{

   public static void main(String[] args){
      System.out.println((((11&19)|5))<<3);
   }
}
```

The program will print out the equation

(((11&19)|5)<<3)

Step 1: 11 & 19

$(11)_{10}$	$=(00001011)_2$	
$(19)_{10}$	$=(00010011)_2$	&
	$`(00000011)_2$	$= (3)_{10}$

Step 2: 3 | 5

$(3)_{10}$	$=(00000011)_2$	
$(5)_{10}$	$=(00000101)_2$	|
	$`(00000111)_2$	$= (7)_{10}$

Step 3: 7 << 3

$(00000111)_2$ << 3 = $(00111000)_2$ = $(56)_{10}$

Therefore, the Java Program outputs 56 when it runs.

*Question c

(c) Describe in your own words the concept of a linked list, using examples and diagram as appropriate. Explain how you would design an algorithm to delete every third link in a singly linked list. Show the operations that would be required. [8 marks]

CS210-2019-January

```
public class Q4_C {
    public static void main(String[] args) {
       SinglyLinkedList list = new SinglyLinkedList();

        // Adding nodes to the list
       Node current = list.head;
       for(int i=1; i<=10; i++) {
          Node newNode = new Node(i);
           if (list.head == null) {
             list.head = newNode;
           } else {
               Node temp = list.head;
               while (temp.next != null) {
                   temp = temp.next;
               }
               temp.next = newNode;
           }
       }

        System.out.println("Original List:");
        list.printList();

        // Delete every third node
        list.deleteEveryThirdNode();

        System.out.println("List after deleting every third
node:");
        list.printList();
    }

}

class Node {
    public int data;
    public Node next;

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

class SinglyLinkedList {
```

CS210-2019-January

```
    Node head;

    public SinglyLinkedList() {
       head = null;
    }

    // Method to delete every third node in the linked list
    public void deleteEveryThirdNode() {
        if (head == null || head.next == null || head.next.next
== null) {
            // If the list is empty or has fewer than three
nodes, no deletion is needed
            return;
        }

        Node currentNode = head;
        Node previousNode = null;
        int count = 1;

        while (currentNode != null) {
            if (count == 3) {
                // Delete the current node
                previousNode.next = currentNode.next;
                count = 1;  // Reset the count after deletion
            } else {
                count++;
            }

            previousNode = currentNode;
            currentNode = currentNode.next;
        }
    }

    // Method to print the linked list
    public void printList() {
        Node currentNode = head;
        while (currentNode != null) {
            System.out.print(currentNode.data + " ");
            currentNode = currentNode.next;
        }
        System.out.println();
    }
}
```

CS210-2019-January

*Question d

(d) An increasing number of technology companies are investigating the development of blockchain technologies. Explain in your own words what blockchain is, and discuss the novel applications it might support in the future [5 marks]

Blockchain technology is a digital ledger technology. Each block contains a set of transactions, and once a block is completed, it is added to the chain in linear chronological order. Blockchain is distributed across a network of computers, making it highly transparent and secure.

Novel Applications that BlockChain may support in the future:

1. Smart Contracts:
2. Supply Chain Management:
3. Healthcare Record Management:
4. Voting Systems:
5. Decentralized Finance (DeFi):
6. Identity Verification:
7. Energy Trading:
8. Intellectual Property and Royalties:
9. Education and Academic Credentials:
10. Internet of Things (IoT):