Question 1

[25 marks]

1 2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder. Write a Java program that outputs the smallest positive number that is evenly divisible by all of the numbers from 1 to 20. Include comments that explain your code clearly. [25 marks]

```
public class Q1 {
    public static void main (String args[]) {
        int noRemainder = 1;
        // Aggregate the LCM from 1 to 20.
        for (int i = 1 ; i <= 20; i++) {
            noRemainder = lcm (noRemainder, i);
        }
        System.out.println(noRemainder);
    }

    // Greatest Common Divisor
    public static int gcd (int a, int b) {
        if (b == 0) {
            return a;
        }
        return gcd(b, a % b);
    }

    // Least Common Multiple
    // LCM(a,b) = a X b / GCD(a,b)
    public static int lcm (int a, int b) {
        return (a * b) / gcd(a, b);
    }

}
```

Question 2

[25 marks]

2 Two integers are coprime if the only positive integer that divides into both of them is 1. In other words, the greatest common divisor of two coprime numbers is 1. Write a Java program that uses a Monte Carlo simulation to determine the probability that two randomly selected numbers will be coprime with each other. Include comments that explain your code clearly. [25 marks]

```
public class Q2 {
    public static void main (String args[]) {
        // Total number of Monte Carlo
        int N = 1000000;
        int x,y;
        int sum = 0;
        for (int i = 0; i < N; i++) {
            x = (int) (Integer.MAX_VALUE * Math.random());
            y = (int) (Integer.MAX_VALUE * Math.random());
            // Coprime number of Monte Carlo
            if (gcd(x,y) == 1) sum++;
        }
        double p =  (double)100 * sum / (double) N;
        // Keep 2 decimal digits
        System.out.println(String.format("%.2f", p) + "%");
    }

    // Greatest Common Divisor
    // If gcd(a,b) == 1, then a and b are coprime.
    public static int gcd (int a, int b) {
        if (b == 0) {
            return a;
        }
        return gcd(b, a % b);
    }

}
```

Question 3

[25 marks]

3 Write a Java program that manipulates a priority queue according to given insert and remove commands, and then outputs the string that is in the middle of the priority queue. Include comments that explain your code clearly. [25 marks]

If there is an even number of strings in the queue, thus two middle strings, the program should output the one which is nearest the front. If a remove command is issued for an empty queue nothing should happen.

The way that priority is settled is by the number of vowels in a word. If a word has more vowels, then it has higher priority. If two words have the same number of vowels, then the higher priority word is the one that has been in the queue the longest.

Sample Input
INSERT this
INSERT is
INSERT how
INSERT to
REMOVE
INSERT do
REMOVE
INSERT it

Sample Output
to

```
import java.util.Scanner;
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedList;

public class Q3_linkedList {
    public static void main (String args[]) {
        Scanner scanner = new Scanner(System.in);
        LinkedList <String> priorityQueue = new LinkedList<String>();
        while (true) {
            String userInput = scanner.nextLine();
        //If input includes "INSERT" at first, add element to the queue.
            if(userInput.split(" ")[0].toUpperCase().equals("INSERT")) {
             priorityQueue.add(userInput.split(" ")[1]);
            }
```

CS210-2022-January

```
            //If input is "REMOVE", sort the queue first and remove the
            first element.
            if(userInput.toUpperCase().equals("REMOVE")) {

            Collections.sort(priorityQueue, (new Comparator<String>() {
                @Override
                public int compare(String o1, String o2) {
                    // 1 - More Vowel has higher priority
                    if(countVowels(o2)!=countVowels(o1)) {
                        return countVowels(o2) - countVowels(o1);
                    }
                    // 2 - If same, longer word has higher priority
                    else if (o2.length()!= o1.length()) {
                        return o2.length() - o1.length();
                    }
                    // 3 - else, remain the input sequence
                    else {
                        return 1;
                    }
                }

                //Method: Compare the Vowels
                public static int countVowels(String str) {
                    int count = 0;
                    for (int i = 0; i < str.length(); i++) {
                        char ch = Character.toLowerCase(str.charAt(i));
                        if (ch == 'a' || ch == 'e' || ch == 'i'
                                || ch == 'o' || ch == 'u') {
                            count++;
                         }
                    }
                    return count;
                }
            }));
// Remove command is issued for an empty queue,  nothing should happen.
            if(!priorityQueue.isEmpty()) priorityQueue.poll();
            }
            // If input empty String, close the scanner.
            if (userInput.isEmpty()) {
            scanner.close();
            break;
            }
            // Make a new input line, ignore other invalid input.
        }
```

```
        // Get the middle string of priority queue.
        int size = priorityQueue.size();
        String middleElement;
        for (int i = 0; i < size / 2 - 1; i++) {
        priorityQueue.poll();
        }
        // Even number -> the middle two -> nearest the front
        if(size % 2 == 0) {
        middleElement = priorityQueue.peek();
        // Odd number -> the middle one
        } else {
        priorityQueue.poll();
        middleElement = priorityQueue.peek();
        }
        System.out.println(middleElement);
    }

}
```

CS210-2022-January

Question 4

[25 marks]

4 Write a Java program that sorts a long list of words using a special ordering. The first input line is an int representing the total number of words, followed by a word on each line. All the words should be sorted by the total sum of all the ASCII values of each character in the word. The words that are outputted first should be those with the lowest ASCII sum. For example, the ASCII value for 'a' is 97, while that for 'n' is 110, so the word "an" has an ASCII sum of 207. Because this total is relatively low, “an” should be one of the first words to be outputted. [25 marks]

Words that have the exact same ASCII sum (e.g., "dog" and "god") should be sorted in REVERSE alphabetical order (so "god" would be outputted before "dog" since it's nearer the end of the dictionary).

State the complexity of your program using **Big O notation**. The more efficient it is, the better. Include comments that explain your code clearly.

Sample Input

10
one
two
three
four
five
six
seven
eight
nine
ten

Sample Output

one
ten
six
two
nine
five
four
eight
three
seven

```
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedList;
import java.util.Scanner;
```

CS210-2022-January

```java
public class Q4 {
    public static void main (String args[]) {
        LinkedList <String> input = new LinkedList<String>();
        Scanner sc = new Scanner(System.in);

        // Input total number of words
        int SIZE = sc.nextInt();
        sc.nextLine(); // change a new line to avoid bug
        // Input each word
        for (int i = 0; i < SIZE; i++) {
            input.add(sc.nextLine());
        }
        sc.close();

        // Sort the List after input.
        Collections.sort(input, (new Comparator<String>() {
            @Override
            public int compare(String o1, String o2) {
                // lower sum of ASCII has priority
                if(sumOfASCLL(o1) != sumOfASCLL(o2)) {
                    return sumOfASCLL(o1) - sumOfASCLL(o2);
                }
                // REVERSE alphabetical order
                else {
                    return o2.compareTo(o1);
                }
            }

            // Method: Calculate the total sum of the ASCII characters
            public int sumOfASCLL (String str) {
                int sum = 0;
                for(int i = 0; i < str.length(); i++) {
                    sum = sum + (int)str.charAt(i);
                }
                return sum;
            }
        }));

        // Print out each element in the LinkedList
        for (String element : input) {
            System.out.println(element);
        }
    }
}
```

CS210-2022-January

【Question】 State the complexity of your program using Big O notation.

【Answer】 The time complexity in my program is (N*log (N))
Because if the element amount in Collection.sort() is very high,
Java will use MergSort to change the order.

In merge sort, if you want to use BinaryTree to divide N elements into one element, you need log2(N) times.
In each layer, every element should compare each other for N time.
Therefore the Big O Notation for mergesort is O(N*log(N)).

```
System.out.println(sumOfASCLL("one")); //322
System.out.println(sumOfASCLL("ten")); //327
System.out.println(sumOfASCLL("six")); //340
System.out.println(sumOfASCLL("two")); //346
System.out.println(sumOfASCLL("nine")); //426
System.out.println(sumOfASCLL("five")); //426
System.out.println(sumOfASCLL("four")); //444
System.out.println(sumOfASCLL("eight")); //529
System.out.println(sumOfASCLL("three")); //536
System.out.println(sumOfASCLL("seven")); //545
```

CS210-2022-January