Question 1

1. Write a Java program that uses a Monte Carlo simulation to calculate the probability that two randomly selected numbers are coprime. [25 marks]

Two integers are said to be coprime if the only positive integer that divides both of them is 1. For example, 12 and 49 are coprime, because the only positive integer that divides evenly into both of them is 1. In contrast, 12 and 21 are not coprime, because 3 divides into both of them.

Include comments which explain your code clearly.

```
public class Q1 {
    public static void main (String args[]) {
        // Total number of Monte Carlo
        int N = 1000000;
        int count = 0;
        for (int i = 0; i < N; i++) {
            int n1 = (int) (Integer.MAX_VALUE * Math.random());
            int n2 = (int) (Integer.MAX_VALUE * Math.random());
            // Coprime number of Monte Carlo
            if(gcd(n1,n2) == 1) count++;
        }
        double probability = (double) 100 * count / (double) N;
        // Keep 2 decimal digits
        System.out.println(String.format("%.2f", probability) + " %");
    }

    // gcd = 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;
        }
        else {
            return gcd(b, a % b);
        }
    }
}
```

Question 2

2. Write a Java program that takes in a String array of 10,000 English words, and sorts them according to the number of unique letters they contain, before printing them out in order. The words containing the greatest number of unique letters should be printed out first, followed by those containing fewer unique letters. If two words use the same number of unique letters, then they should be sorted lexicographically (i.e. like a dictionary). [25 marks]

For example, the word "*fantastic*" uses 7 unique letters, so it should be printed out before "*banana*", which uses only 3 unique letters.

Include comments which explain your code clearly.

State the Big O complexity of your program and justify your reasoning.

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

public class Q2 {
    public static void main (String args[]) {
        Queue<String> pq = new PriorityQueue<String> ( new Comparator<String>() {
            @Override
            public int compare (String o1, String o2) {
                // 1 - more unique letters first
                if(countUniqueChar(o1) != countUniqueChar(o2)) {
                    return (countUniqueChar(o2)-countUniqueChar(o1));
                }
                // 2 - If unique letter is same -> lexicographically
                else {
                    return o1.compareTo(o2);
                }
            }

            // Use Set to calculate the number of unique letter.
            // Set can filter the duplicate letter.
```

```
            public int countUniqueChar(String input) {
                Set<Character> uniqueChar = new HashSet<Character>();
                for(char c : input.toCharArray()) {
                    uniqueChar.add(c);
                }
                return uniqueChar.size();
            }
        }); // end of Queue and its Comparator

        Scanner sc = new Scanner(System.in);
        int SIZE = 10000;
        for (int i = 0; i < SIZE; i++) {
            pq.add(sc.nextLine());
        }
        sc.close();
        // Print out all sorted String in PriorityQueue
        while(!pq.isEmpty()) {
            System.out.println(pq.poll());
        }
    }
}
```

CS210-2020-January

Question 3

3. Write a Java program that manipulates a queue according to the given insert and remove commands, and then outputs the string that is in the middle of the queue. If there is an even number of strings in the queue, thus two middle strings, output the one which is nearest the front. If a remove command is issued for an empty queue then nothing should happen. [25 marks]

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

Output Format
Output the string that is in the middle of the queue following the given commands. If there are two middle strings, output the one nearest the front. If the queue is empty output "empty".

Sample Input

INSERT these
INSERT words
INSERT go
INSERT in
REMOVE
INSERT the
REMOVE
INSERT queue

Sample Output
in

Include comments which explain your code clearly.

Your answer should provide the full queue class.

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

public class Q3 {
    public static void main (String args[]) {
        Queue<String> queue = new LinkedList<String>();
        Scanner sc = new Scanner(System.in);

        while(true) {
            String inputLine = sc.nextLine();
            // INSERT Command
            if(inputLine.split(" ")[0].toUpperCase().equals("INSERT")) {
                queue.add(inputLine.split(" ")[1]);
            }
```

```
            // REMOVE Command
            if(inputLine.toUpperCase().equals("REMOVE")) {
                /* If a remove command is issued for an
                   empty queue, then nothing should happen. */
                queue.poll();
            }
            // End inputs with an empty string
            if(inputLine.isEmpty()) {
                sc.close();
                break;
            }
        }

        // Output the string in the middle of the queue
        int SIZE = queue.size();
        for (int i = 0; i < SIZE/2 -1; i++) {
            queue.poll();
        }
        // Two middle String, output the one nearest the front
        if (SIZE % 2 == 0) {
            System.out.println(queue.peek());
        }
        // One middle String, output that middle one.
        else {
            queue.poll();
            System.out.println(queue.peek());
        }
    }

}
```

Question 4

Question 4-a

4. a) Analyse the Big O complexity of the following piece of Java code, and explain your reasoning.

```
for(int i = 3; i< n*20; i++){
    for(int j = 10; j >= 0; j--){
      for(int k = j; k < n; k++){
            counter++;
      }
    }
}
```

Big O complexity – $O(n^2)$

Input variable -> n

Loop 1: `for(int i = 3; i< n*20; i++)` **-> O(n)**

Loop 2: `for(int j = 10; j >= 0; j--)` **-> O(1)**

Loop 3: `for(int k = j; k < n; k++)` **-> O(n)**

Considering nested loops, the time complexity is $O(n \times 1 \times n) = O(n^2)$.

Question 4-b

b) What does the following recursive method return given an input of 7? Explain your reasoning. Assume that `f[]` is a class variable initialized with zeroes. [4 marks]

```
public int f(int n) {
    if(n==0) return 1;
    if(n==1) return 1;
    if(f[n]!=0){
        return f[n];
    }else{
        f[n] = f(n-1) + f(n-2);
        return f[n];
    }
}
```

$f[0]=1$ (base case)
$f[1]=1$ (base case)
$f[2]=f[1]+f[0]=1+1=2$
$f[3]=f[2]+f[1]=2+1=3$
$f[4]=f[3]+f[2]=3+2=5$
$f[5]=f[4]+f[3]=5+3=8$
$f[6]=f[5]+f[4]=8+5=13$
$f[7]=f[6]+f[5]=13+8=21$

CS210-2020-January

Question 4-c

c) What does the following Java program output when run? Show clearly how the answer is derived. [4 marks]

```
public class BitManipulation{

   public static void main(String[] args){
      System.out.println(((32|13)>>2)&9);
   }
}
```

Step 1: 32 | 13

$(32)_{10}$	$=(00100000)_2$	
$(13)_{10}$	$=(00100000)_2$	I
	$`(00101101)_2$	$= (45)_{10}$

Step 2: 45 >> 2

$(00101101)_2 >> 2 = (00001011)_2 = (11)_{10}$

Step 3: 11 & 9

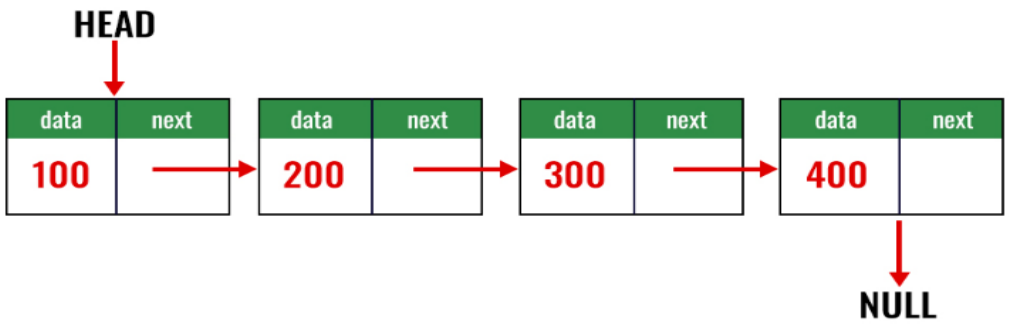
$(11)_{10}$	$=(00001011)_2$	
$(9)_{10}$	$=(00001001)_2$	&
	$`(00001001)_2$	$= (9)_{10}$

Therefore, the Java Program outputs 9 when run.

Question 4-d

d) 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 the tail of a double ended singly linked list.

A linked list is a linear data structure consisting of a sequence of elements, each element contains a value and a reference (or a pointer) to the next element in the sequence.



[Method : delete the tail node of the linked list]

```
public void deleteTail() {
    // If the list is empty, nothing to delete
    if (head == null) {
        return;
    }

    // If the list has only one node, set head to null
    if (head.next == null) {
        head = null;
        return;
    }

    // Traverse the list to find the tail's previous node
    Node currentNode = head;
    while (currentNode.next.next != null) {
        currentNode = currentNode.next;
    }

    // Update the previous node's next pointer to null to remove the tail node
    currentNode.next = null;
}
```

Question 4-e

e) Show step by step how the numbers below would be sorted by

i) Insertion sort
ii) Merge sort
iii) Selection sort

35 86 10 58 26 96 38 52

i) Insertion Sort

[35 ,86, 10, 58, 26, 96, 38, 52]

insert 35, sorted list: **[35]**
insert 86, sorted list: **[35, 86]**
insert 10, sorted list: **[10, 35, 86]**
insert 58, sorted list: **[10, 35, 58, 86]**
insert 26, sorted list: **[10, 26, 35, 58, 86]**
insert 96, sorted list: **[10, 26, 35, 58, 86, 96]**
insert 38, sorted list: **[10, 26, 35, 38, 58, 86, 96]**
insert 52, sorted list: **[10, 26, 35, 38, 52, 58, 86, 96]**

Finally, sorted list: [10, 26, 35, 38, 52, 58, 86, 96]

CS210-2020-January

ii) Merge Sort

[35 ,86, 10, 58, 26, 96, 38, 52]

- Divide the list: 35, 86, 10, 58, 26, 96, 38, 52
- Merge pairs and sort: [35, 86], [10, 58], [26, 96], [38, 52]
- Merge sublists and sort: [26, 38, 52, 96], [26, 38, 52, 96]
- Merge the two sorted sublists: [10, 26, 35, 38, 52, 58, 86, 96]

Finally, sorted list: [10, 26, 35, 38, 52, 58, 86, 96]

iii) Selection Sort

[35 ,86, 10, 58, 26, 96, 38, 52]

- Select min 10, swap with 35: [10, 86, 35, 58, 26, 96, 38, 52]
- Select min 26, swap with 86: [10, 26, 35, 58, 86, 96, 38, 52]
- Select min 35, no swap:
- Select min 38, swap with 58, [10, 26, 35, 38, 86, 96, 58, 52]
- Select min 52, swap with 86: [10, 26, 35, 38, 52, 96, 58, 86]
- Select min 58, swap with 96: [10, 26, 35, 38, 52, 58, 58, 96]
- Select min 86, no swap:
- Select min 96, no swap:

Finally, sorted list: [10, 26, 35, 38, 52, 58, 86, 96]