Question 1

[20 marks]

1 Write a Java program given the following specification and provide comments which explain how your algorithm works. Estimate the **Big O complexity** of your program and explain your reasoning clearly.

Problem Statement
The goal is to read in a list of integers into an array and output the one which is closest to the average. For example, the average of 4, 8, 2, 3 and 5 is 4.25, and 4 is the closest to this. If there are two values equally close then choose the one which was earliest in the list.

Input Format
The first line contains *N*, the number of inputs. The second line contains *N* integers separated by a space.

Output Format
A number that represents the value that is closest to the average.

Constraints
$1 \leq N \leq 1000$
$-1000 \leq A[i] \leq 1000$

Sample Input
5
36
73
26
85
83

Sample Output
73

```
import java.util.Scanner;


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

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

CS210-2017-January

```
        // Input the number and calculate their average
        int intArray [] = new int [N];
        double sum = 0.0;
        double average = 0.0;
        for (int i = 0 ; i < N ; i++) {
            int input = Integer.parseInt(sc.nextLine());
            intArray [i] = input;
            sum = sum + (double) input;
        }
        average = sum / (double) N;

        // Find the number closest to the average
        double minDifference = Integer.MAX_VALUE;
        int outputElement = 0;
        for (int i = 0 ; i < N ; i++) {
            double difference = Math.abs(average -
            (double)intArray[i]);
    /* If two values are equally close to the average, choose the one  which
was earliest in the list, so we ignore '=' in the case . */
            if(difference < minDifference) {
                outputElement = intArray[i];
                minDifference = difference;
            }
        }

        System.out.println(outputElement);
    }
}
```

[Big O complexity] - O(n)

I use two for loops in the program,
each loop is related to the input number N;
So the complexity is O(N) + O（N） = O(N) ;

CS210-2017-January

Question 2

[20 marks]

2 Write a Java program given the following specification and provide comments which explain how your algorithm works. Estimate the **Big O complexity** of your program and explain your reasoning clearly.

Problem Statement
The goal is to read in the quantity of numbers used in a lottery, the quantity of numbers removed in each draw, and output the probability that the numbers will be drawn in ascending order (e.g. 4, 18, 24, 25, 42).

Input Format
The first line contains *N*, the quantity of numbers used in the lottery (e.g. the EuroMillions lottery uses 50 numbers). The second line contains *D*, the quantity of numbers drawn in a lottery draw (e.g. 5 random numbers are drawn in each EuroMillions draw).

Output Format
An integer from 0 to 100 representing the percentage probability that the numbers will be drawn in ascending order.

Constraints
$0 \leq N \leq 1000$
$0 \leq D \leq 100$

Sample Input
100
2

Sample Output
50

```java
import java.util.Scanner;
import java.util.Set;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.ArrayList;

public class Q2 {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        // Input the quantity number used in the lottery
        int N = Integer.parseInt(sc.nextLine());
```

CS210-2017-January

```
        // Input the quantity number drawn in a lottery draw
        int D = Integer.parseInt(sc.nextLine());
        sc.close();

        // Use Monte Carlo Model
        int ALL = 1000000;
        int count = 0;

        MONTECARLO:
        for (int i = 0; i < ALL; i++) {
            //Create a new drawbox and put it into Set.
            Set<Integer> uniqueNumbers = new LinkedHashSet<Integer>();
             // Avoid duplicate and remember the input sequence.
            while(uniqueNumbers.size() < D) {
                // Select one number from Lottery [1,2....,N]
                int randomNumber = (int) (N * Math.random() + 1);
                uniqueNumbers.add(randomNumber);
            }

            // Check whether the drawn numbers are ASCending
            List<Integer> list = new ArrayList<>(uniqueNumbers);
             for (int j = 1; j < list.size(); j++) {
                // Not ascending, continue to the next Monte Carlo
                 if (list.get(j - 1) > list.get(j)) {
                    continue MONTECARLO;
                 }
             }
            // Ascending, count a time
            count++;
        }

        // Calculate the probability into percentage
        double p = (double)(count * 100)/ (double) ALL;
        // Round up to the nearest Integer
        System.out.println(Math.round(p));
    }
}
```

[Big O complexity] - $O(n^2)$

I use two for loops in the program. The outer loop is related to ALL -- the simulation times of Monte Carlo Simualtion. The inner loop is related to D -- the quantity number drawn from lottery.

So the complexity is O(N) * O（N） = $O(N^2)$;

CS210-2017-January

Question 3

3 Write a Java program given the following specification and provide comments which explain how your algorithm works. Estimate the **Big O complexity** of your program and explain your reasoning clearly.

Problem Statement
The goal is to sort a list of words by the character in each word that comes earliest in the alphabet (e.g. all words containing an 'a', then all words containing a 'b' but no 'a', then all words containing a 'c' but no 'a' or 'b' etc.). To separate words with the same earliest character, defer to the second earliest character and then the third etc. (e.g. both 'ant' and 'apple' have an 'a', but 'apple' should come before 'ant' because its second earliest character is an 'e', whereas that of 'ant' is an 'n'). If two words use all the same characters (e.g. 'babble' and 'able') then use alphabetical ordering to separate them.

Input Format
The first line contains *N* (number of words) followed by *N* lines. Each line contains a String.

Output Format
A line consisting of the words sorted in alphabetical order, each separated by a space.

Constraints
$1 \leq N \leq 100$

Sample Input
4
one
two
three
four

Sample Output
three one four two

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

CS210-2017-January

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

        // number of words followed by N lines
        int N = Integer.parseInt(sc.nextLine());
        // Each line contains a String
        for(int i = 0; i < N; i++) {
            String inputLine = sc.nextLine();
            words.add(inputLine);
        }
        sc.close();


        Collections.sort(words, new Comparator<String>() {
            @Override
            public int compare (String s1, String s2) {
                String []alphabet = {"a","b","c","d","e","f","g",
                        "h","i","j","k","l","m","n","o","p","q",
                        "r","s","t","u","v","w","x","y","z"};
                // Sort the words by the earliest character
                for (int i = 0; i < 26; i++) {
                    if(s1.contains(alphabet[i])
                    && !s2.contains(alphabet[i]))
                    {
                        return -1;
                    }
                    else if(!s1.contains(alphabet[i])
                    && s2.contains(alphabet[i]))
                    {
                        return 1;
                    }
                    else
                    {
                        continue;
                    }
                }

                // If two words use same characters -> alphabetical order
                return s1.compareTo(s2);
            }
        });    // End of Collection and its Comparator.
```

CS210-2017-January

```
        // Print out the sorted words
        for(String element:words) {
            System.out.print(element + " ");
        }
    }
}
```

[Big O complexity] - O(n * log(n))

1) When I add elements into LinkedList or output the element from the LinkedList, they are related to N -- the number of words, so the time complexity is O(n)

2) When I use Collection.sort(), it will use merge sort if the number of input is very high. The time complexity of merge sort is O(n * log(n)).

Therefore, the time complexity of this program is

O(n) + O(n * log(n)) + O(n) = O(n * log(n))

CS210-2017-January

Question 4

[20 marks]

4 Complete the Java method given the following specification and provide comments which explain how your algorithm works. Estimate the **Big O complexity** of your method and explain your reasoning clearly.

Problem Statement
A loop in a linked list occurs when following the linked list chain brings you back where you started. For example, link 1 points to link 2 which points to link 3 which points to link 1, to link 2, to link 3, to link 1...and you go round in an infinite loop. Each test case here involves a single-ended singly-linked list with a potential loop. Output the size of the loop, or 0 if there is no loop. In the above example, the loop is of size 3 (e.g. link 1, link 2, link 3...repeating forever).

Input Format
The linked list is created automatically by the stub code.

Output Format
Complete the findLoopLength() method below. Output an integer, either 0 if the linked list is empty or has no loop, or else >0 corresponding to the length of the loop in the linked list.

Constraints
0≤#(links)≤100

Stub Code

```
import java.util.*;

public class Solution {
    public static void main(String args[] ){
        Scanner myscanner = new Scanner(System.in);


        int num = Integer.parseInt(myscanner.nextLine());
        Link[] array = new Link[num];
        for(int i=0;i<num;i++){
            array[i]=new Link(myscanner.nextLine());
        }
        while(myscanner.hasNext()){
            int select=myscanner.nextInt();
            int next=myscanner.nextInt();
            if(next!=-1){
                array[select].next=array[next];
            }
        }
        LinkedList mylist = new LinkedList();
        if(num>0){
            mylist.first=array[0];
        }
        System.out.println(findLoopLength(mylist));
    }
```

```
    public static int findLoopLength(LinkedList mylist){
/*return 0 if empty or no loop, otherwise length of loop
in mylist*/
        return 0;
    }
}
```

```
class Link{
    public String data;
    public Link next;

    public Link(String input){
        data=input;
        next=null;
    }
}

class LinkedList {
    public Link first;

    public LinkedList( ){
            first=null;
    }

    public boolean isEmpty( ){
            return (first==null);
    }

    public void insertHead(Link insert){
        if(isEmpty()){
            first=insert;
        }else{
            insert.next=first;
            first=insert;
        }
    }
```

Floyd's cycle-finding Algorithm:
FAST pointer moves 2 steps in one time
SLOW pointer moves 1 steps in one time
If there has a loop, FAST will finally catch up with SLOW.

```
public static int findLoopLength(LinkedList mylist){
      Link slow = mylist.first;
      Link fast = mylist.first;
      boolean hasLoop = false;

      //Find whether has a LOOP
      while(fast != null && fast.next != null){
         slow = slow.next;
         fast = fast.next.next;
         if(slow == fast) {
            hasLoop = true;
            break;
         }
      }
```

CS210-2017-January

```
        // Calculate the length of Loop
        // No LOOP/ Empty List -> default value 0
        int loopLength = 0;
        if (hasLoop) {
            do {
                fast = fast.next;
                loopLength++;
            } while (fast != slow);
        }

        return loopLength;
}
```

[Big O complexity] - O(n)

In the method, there is one while loop.

The loop is related with the num of Link.

int *num = Integer.parseInt(myscanner.nextLine());*

So the Big O Complexity is O(n).

CS210-2017-January