# Data Structures and Algorithms

# INFO 6205

# Homework 3

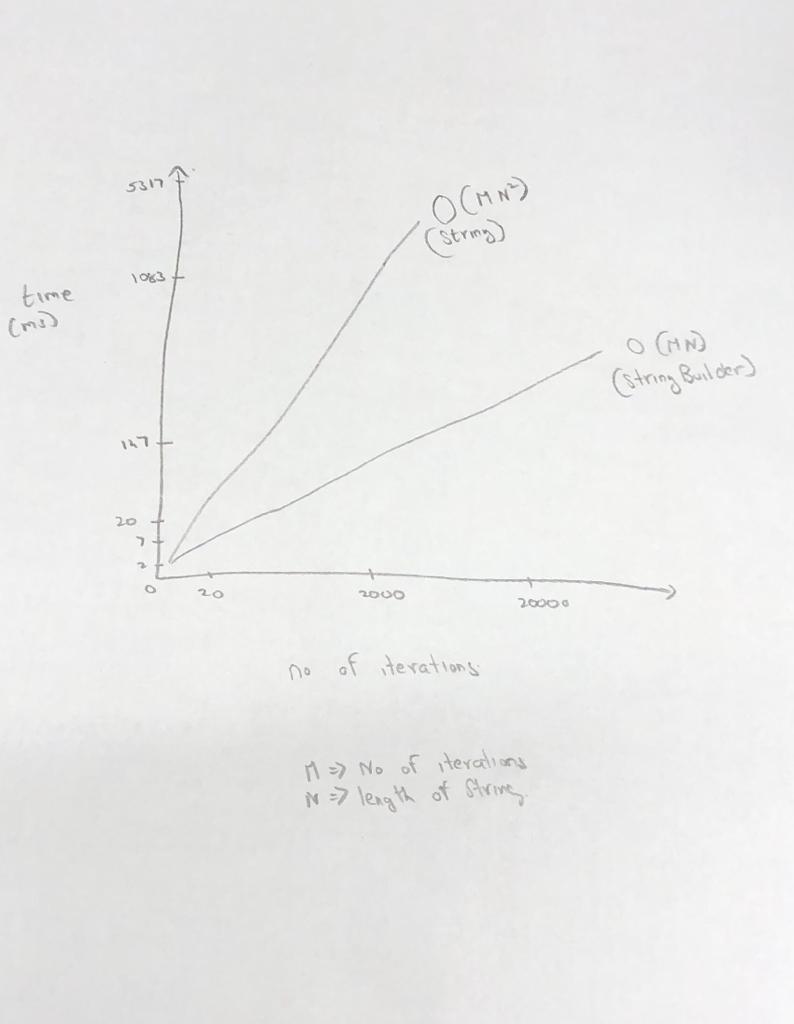
# Due: February 2, 2019

Put all your java, compiled class files and documentation files into a zip file named Homework3.zip and submit it via the drop box on the blackboard before the END of due date. Put your name on all .java files. There will be a short quiz on this homework.

1. Write a Java program that generates random text string with a length of 500 bytes for 200,000 iterations. For each iteration, reverse the string using: a) String operations, and b) StringBuilder operation(s). Then c) What is the running time complexity of (a) and (b) after all iterations?.

**Ans: Running time complexity of O(MN^2)-String and O(MN)- StringBuilder**

d) Present your results in (c) as a graph showing the running times.



2. Consider String “Test is a hard test”.

a) Generate a binary Huffman Tree

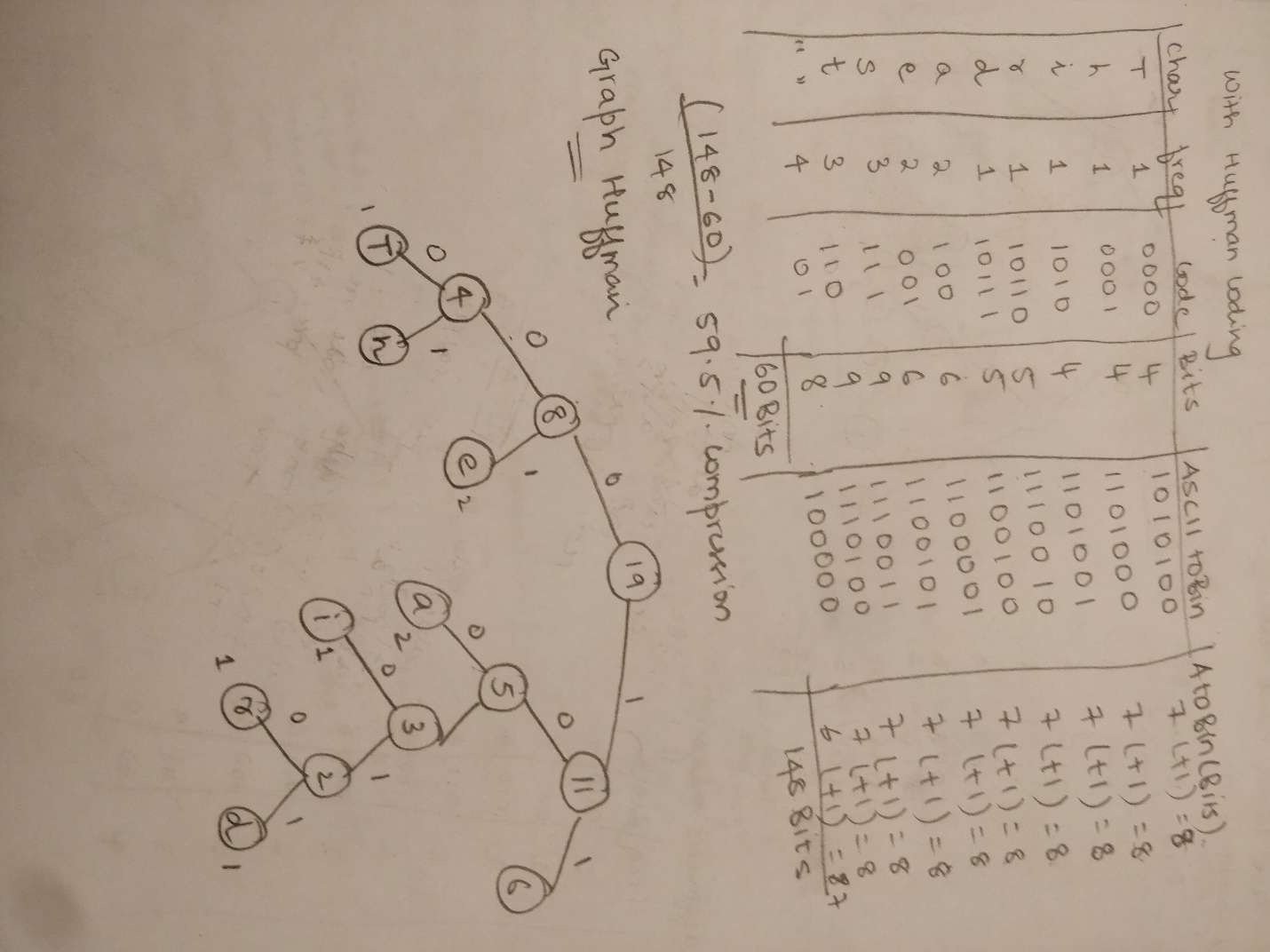
b) Show binary data both before and after compression. Analyze difference.

c) Consider Java code:

https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

Write Pseudo-Code for the Huffman algorithm

d) Compile Java code and run it with the input string provided above.



3. Consider signed byte X, and unsigned byte Y. What are the possible values for both X

and Y can have?

**Ans:** A signed byte can have values from **-128 to 127** and an unsigned byte can have values from **0 to 255**

4. Write Java Factorial program for n! where n=7 and n=14.

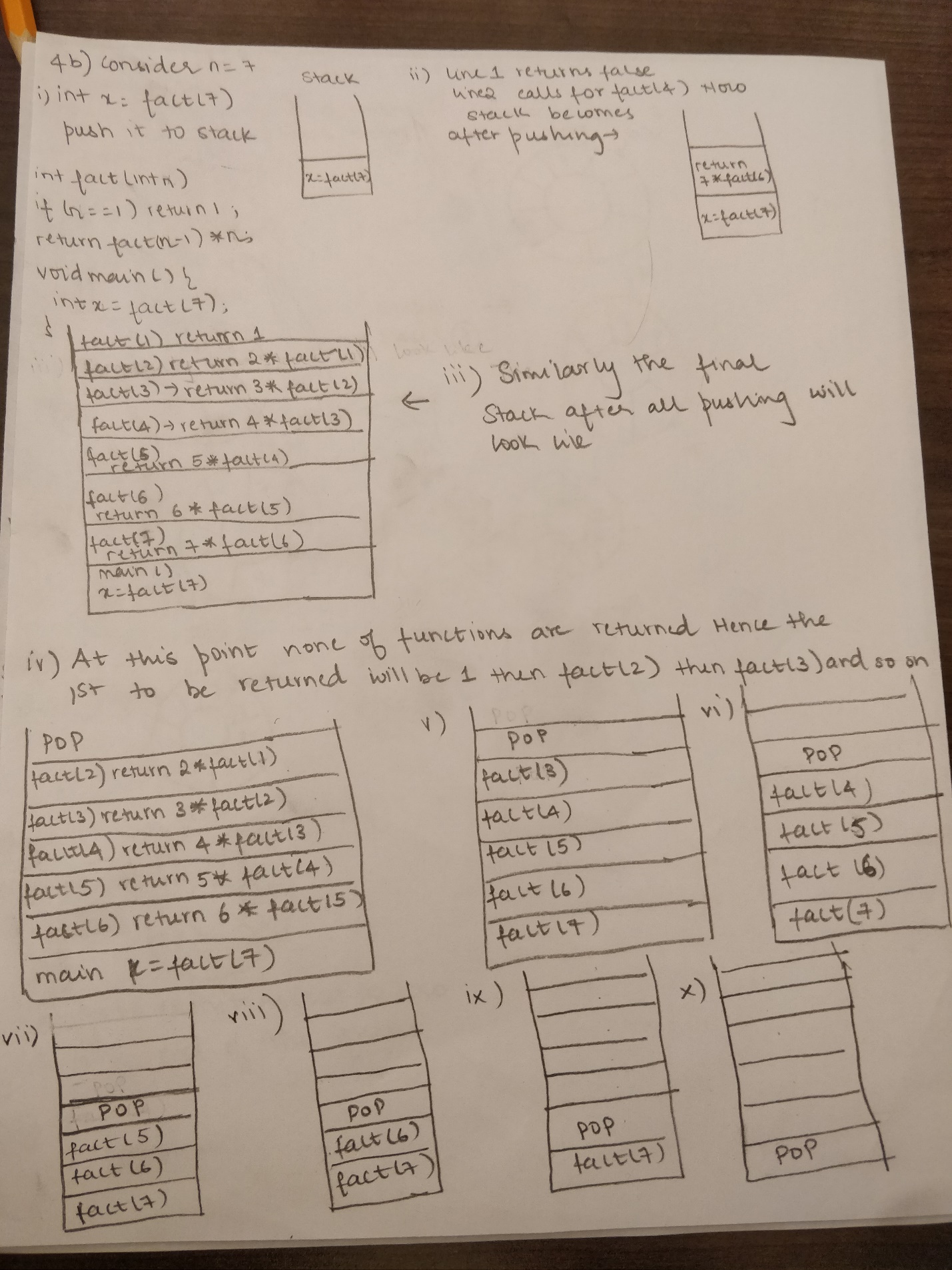
a) Compile the code and run.

b) Step through each recursive call and show the Stack for push and pop. For each

pop operation, show the STATE of push operation (ie: how “n” is being kept track of).

c) What is the Time and Space Complexity of Factorial function?

**Ans:** Time and space complexity is O(n)



5. Java is Pass-by-Value, what does that mean?

**Ans:** It means that the method parameter values are copied to another variable and then the copied object is passed.

Consider the following two programs,

Program-1:

public static void main(String[] args) {

Dog aDog = new Dog(“Bella");

Dog oldDog = aDog;

changeName(aDog);

aDog.getName().equals("Bella"); **True:** **aDog.getName() still holds ‘Bella’**

aDog.getName().equals("Molly"); **False**

aDog == oldDog; **True: Because of line Dog oldDog=aDog;**

}

public static void changeName(Dog d) {

d.getName().equals("Bella"); **True: Ref is of aDog**

d = new Dog("Molly");

d.getName().equals("Molly");

}

Program-2:

public static void main(String[] args) {

Dog aDog = new Dog("Bella");

Dog oldDog = aDog;

changeName(aDog);

aDog.getName().equals("Molly"); **True: changeName function returns ‘Molly’**

aDog == oldDog; **True: Still same reference**

}

public static void changeName(Dog d) {

d.getName().equals(“Bella"); **True: Reference is of aDog**

d.setName("Molly");

}

6. Consider the following, Input Data: {7, 38, 3, 9, 82, 10, 31, 24}

a) Graphically build a Circular queue for input data. Discuss and show the Head

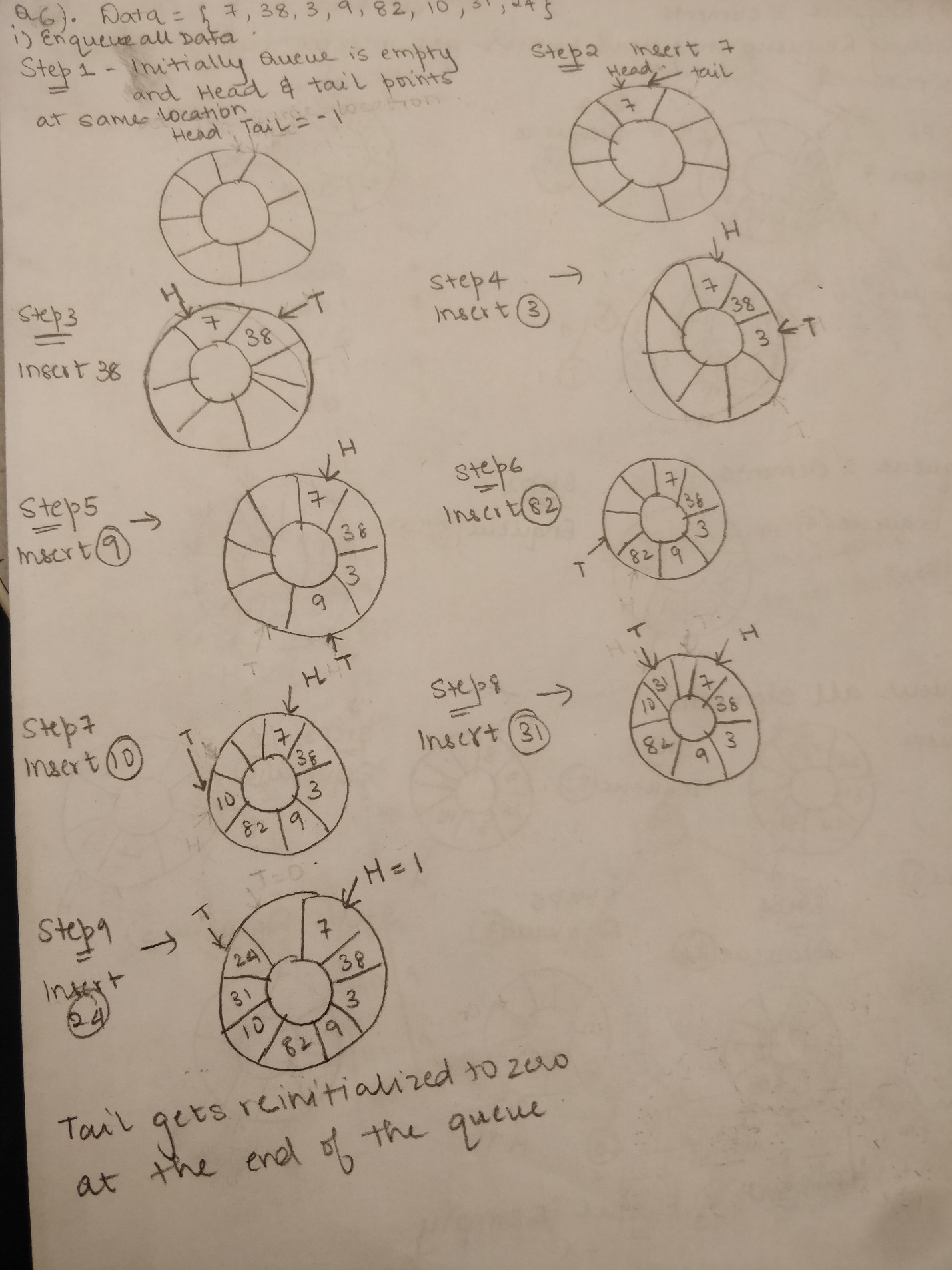
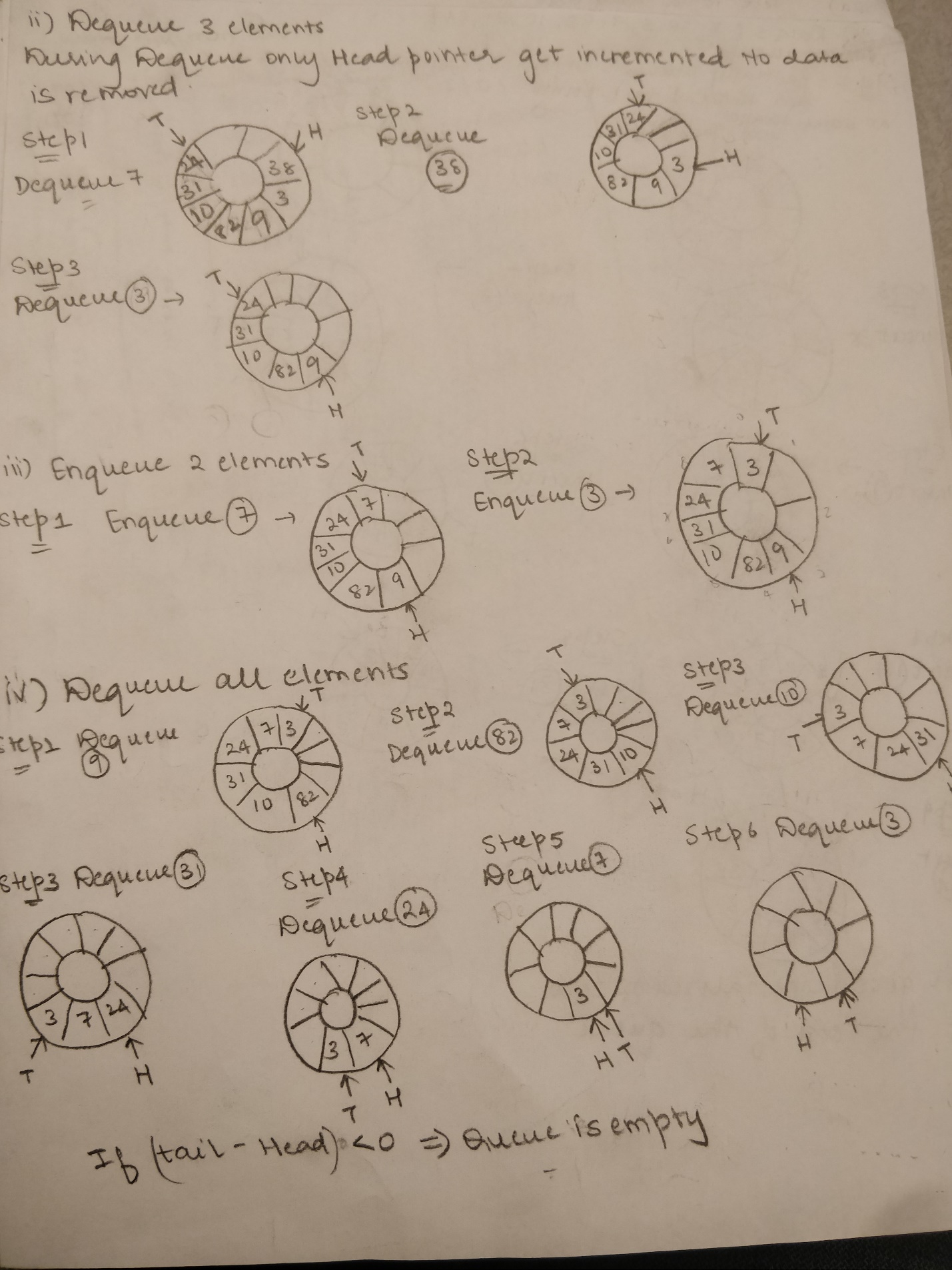
and Tail pointers at each step.

i) enqueue all input data

ii) dequeue three elements

iii) enqueue two elements

iv) dequeue all elements



b) Write Java code for the Circular queue, provide enqueue, dequeue, isEmpty, isFull, and

displayQueue methods, to show the status of the queue with steps described in (a).

Compile code and Run with input data.

http://mooreccac.com/kcppdoc/Recursion.htm