CSC 355. Discrete Structures & Basic Algorithms
Module 3. Introduction to Data Structures

Instructions: Solve the following questions.

Question 1. Written Response.

(a) Write an algorithm that takes a linked list of size N elements and splits it into two linked lists where the first list contains all the elements that are 1 less than or equal to the first element in the list and the second list contains all the elements that are 2 greater than the first element in the list.

(b) Show that your algorithm works by running it on example list given below.

Example:

```
Input: L = 3 → 4 → 1 → 0 → 2 → 9
Output: L1 = 3 → 1 → 0 → 2 and L2 = 4 → 9
```

Question 2. Written Response.

Let *A* be an array of size *N* that is made up of two separate subarrays that are already sorted in increasing order. Let *i* be the index where the first sorted subarray ends. Write an algorithm that runs in *O(N)* time and that merges the two sorted subarrays into a single sorted array.

Example:

```
Input: [0, 2, 5, 6, 2, 6, 6, 7, 8, 9], 3
Output: [0, 2, 2, 5, 6, 6, 6, 7, 8, 9]
```

Question 3. Fill in the blank.

1. Identify the main elements in the Binary Tree described below *(10 points)*:

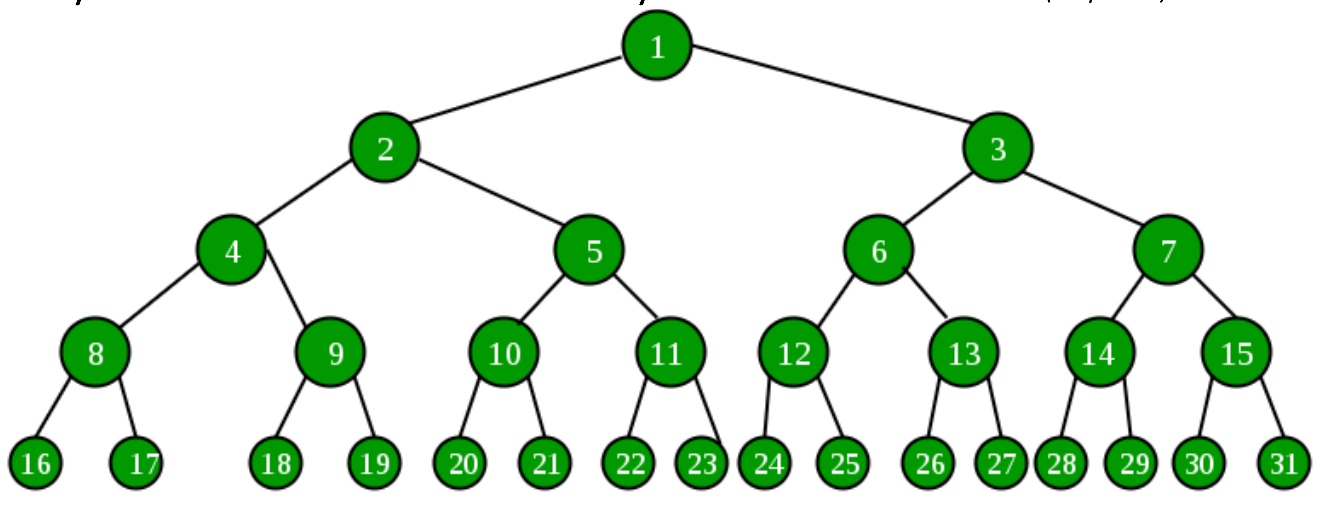


Figure 1. Binary Tree

a) Size of the tree: _______

b) Root node: ______

c) List the internal nodes: ______

d) Identify the child node(s) of node 8 and node 15: ______

e) # Leaf nodes and list them: _______

Question 4. Written Response.

A palindrome is a string that reads the same forwards as backwards. Using only a fixed number of stacks and queues, the stack and queue functions, and a fixed number of int and char variables, write an algorithm to determine if a string is a palindrome. Assume that the string is read from standard input one character at a time. The algorithm should output true or false as appropriate.

Submission Instructions

You must upload your homework in a **pdf** file in the designated area in D2L.

Grading Points

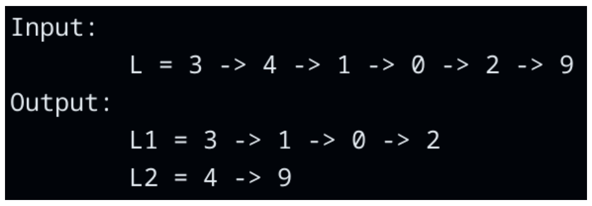
Total Score: 25 points

**Each question has a value of 6.25 points*

Q1)

```java
package Homeworks.hw3;

import java.util.Arrays;
import java.util.Iterator;
import java.util.LinkedList;

public class q1 {
    public static void printList(LinkedList list) {
        Iterator<Integer> iter = list.iterator();
        while (true) {
            System.out.print(iter.next());
            if (iter.hasNext()) {
                System.out.print(" -> ");
                continue;
            }
            return;
        }
    }

    public static void main(String[] args) {
        Integer[] input = { 3, 4, 1, 0, 2, 9 };

        LinkedList<Integer> masterList = new LinkedList<Integer>(Arrays.asList(input));
        LinkedList<Integer> lessThanList = new LinkedList<Integer>();
        LinkedList<Integer> greaterThanList = new LinkedList<Integer>();

        Integer decider = masterList.peekFirst();
        Integer element;
        Iterator<Integer> iter = masterList.iterator();

        while (iter.hasNext()) {
            element = iter.next();

            if (element <= decider) {
                lessThanList.add(element);
            } else {
                greaterThanList.add(element);
            }
        }

        System.out.print("Input:\n\tL = ");
        printList(masterList);

        System.out.println("\nOutput:");
        System.out.print("\tL1 = ");
        printList(lessThanList);

        System.out.print("\n\tL2 = ");
        printList(greaterThanList);

        System.out.println("");

    }
}
```

Result:

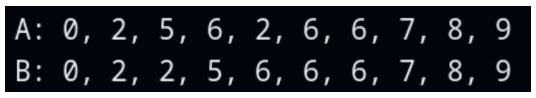


```
Input:
        L = 3 -> 4 -> 1 -> 0 -> 2 -> 9
Output:
        L1 = 3 -> 1 -> 0 -> 2
        L2 = 4 -> 9
```

Q2)

```java
package Homeworks.hw3;

public class q2 {
    public static void main(String[] args) {
        int[] A = { 0, 2, 5, 6, 2, 6, 6, 7, 8, 9 };
        int[] B = new int[A.length];
        int i = 3;
        int j;

        int hold1 = 0;
        int hold2 = i + 1;
        for (j = 0; j < B.length; j++) {
            if (hold2 >= B.length) {
                B[j] = A[hold1++];
                continue;
            } else if (hold1 >= i + 1) {
                B[j] = A[hold2++];
                continue;
            }

            if (A[hold1] <= A[hold2]) {
                B[j] = A[hold1++];
            } else {
                B[j] = A[hold2++];
            }
        }

        // Printing arrays
        System.out.print("A: ");
        for (j = 0; j < A.length - 1; j++) {
            System.out.print(A[j] + ", ");
        }
        System.out.println(A[j]);

        System.out.print("B: ");
        for (j = 0; j < B.length - 1; j++) {
            System.out.print(B[j] + ", ");
        }
        System.out.println(B[j]);
    }
}

```

Result:



```
A: 0, 2, 5, 6, 2, 6, 6, 7, 8, 9
B: 0, 2, 2, 5, 6, 6, 6, 7, 8, 9
```

Q3)

a) Size of the tree: 31

b) Root node: 1

c) List the internal nodes: Nodes 1 - 15

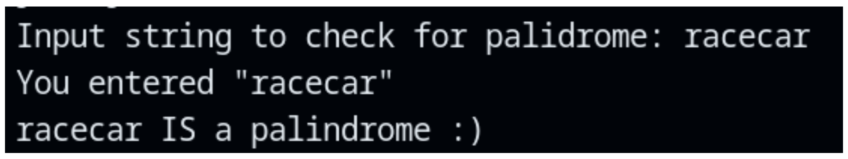
d) Identify the child node(s) of node 8 and node 15: 8 -> 16, 17
15-> 30, 31

e) # Leaf nodes and list them: 16 Leaf Nodes:
Nodes 16 - 31

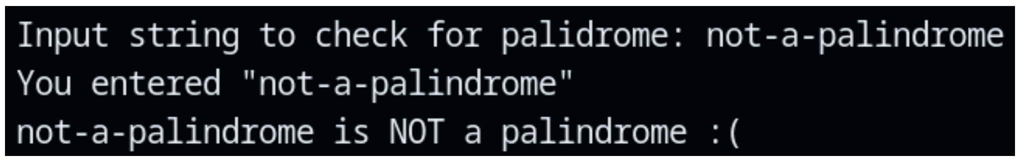
Q4)

```java
package Homeworks.hw3;

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

public class q4 {
    public static void main(String[] args) {
        // Initialize Variables
        Scanner scanner = new Scanner(System.in);
        boolean isPalindrome = true;
        String stringIn;
        char[] charArray;

        Queue<Character> forwards = new LinkedList<Character>();
        Queue<Character> backwards = new LinkedList<Character>();

        // Get string input and convert it to char array
        System.out.print("Input string to check for palidrome: ");
        stringIn = scanner.nextLine();
        charArray = stringIn.toCharArray();
        System.out.printf("You entered \"%s\"\n", stringIn);

        // populate queues for comparison
        for (int i = 0; i < charArray.length; i++) {
            forwards.offer(charArray[i]);
            backwards.offer(charArray[charArray.length - 1 - i]);
        }

        // simultanously poll the qeues while comparing them, flagging false with fail
        // condition
        while (forwards.peek() != null) {
            if (forwards.poll() != backwards.poll()) {
                isPalindrome = false;
                System.out.printf("%s is NOT a palindrome :(\n", stringIn);
                System.exit(0);
            }
        }

        System.out.printf("%s IS a palindrome :)\n", stringIn);
    }
}
```

Results:



```
Input string to check for palidrome: racecar
You entered "racecar"
racecar IS a palindrome :)
```



```
Input string to check for palidrome: not-a-palindrome
You entered "not-a-palindrome"
not-a-palindrome is NOT a palindrome :(
```