Group Member 1 Roll#_____ Group Member 2 Roll#_____ Group Member 3 Roll#_____

Group Member 1 Name_____ Group Member 2 Name_____ Group Member 3 Name_____

Date: 27 / 11 /2023 **Data Structures Lab- Hackathon** **BCS – 3J**

Instructions:

- **Duration:** 1 hour 15 mins
- **Paper consists of 3 questions.**
- **Not following any instructions may result in deduction of marks.**
- **Time and Space complexity should be considered in all solutions.**
- **Understanding the questions is part of the exam.**
- **Return the question paper after the exam.**
- **There are three (3) questions in total. All carry equal marks.**
- **Upload your submission on the google classroom as a compressed file. The naming convention should be: <Id1, Id2, Id3> DS Hackathon**

Question 1

Develop and implement the following operations for a circular linked list:

Create a circular linked list with N nodes, each containing a unique hexadecimal string.
Implement a function to split the circular linked list into two equal parts, handling odd-length cases.
Write a function to delete every Kth node until one node remains and print that node (Josephus problem).
Implement a function to detect whether the list is a palindrome.
Implement a function to reverse every block of M nodes in the list.
Print all nodes in ascending order based on the lexicographical value of the data.

Question 2

Build a BST where each node's data is generated from a given mathematical function f(x).
Implement a function to count the number of nodes with prime numbers as data.
Write a function to find the longest path from the root to any leaf and print the nodes on that path.
Implement a function to mirror the BST and display both the original and mirrored trees.
Write an algorithm to remove all nodes that do not have exactly one child and maintain BST properties.
Implement a function that checks whether the BST forms a valid max-heap.

Question 3

Develop an AVL tree:

Insert customer data with strict rules (e.g., names with vowels at even positions).
Implement a function that performs a complex search based on partial matches, returning all names matching a given pattern.
Implement a function that finds and displays all nodes that have heights differing by more than one from their siblings.
Write a function that deletes all nodes whose names have more than three vowels and maintains AVL balance.
Implement a function that rotates subtrees to balance based on a given threshold.
Print all nodes in post-order traversal along with their balance factor.