# Assignment-6 for Data Structure and Integration in Program

Subject: CSW2 (CSE 3141)
Session: Jan to May 2025
Branch: CSE
Section: All
Course Outcomes: CO3

Learning Levels: Remembering (L1), Understanding (L2), Application (L3), Analysis (L4), Evaluation(L5), Creation (L6)

| Q no. | Questions | Learning Levels |
|---|---|---|
| Q1. | Design a class **BSTNode** in Java with a member variable **info** to store an integer and two references, **left** and **right**, to represent its left and right children. Implement a constructor to initialize these attributes. Develop a method to insert a node while maintaining the properties of a **binary search tree**. Extend this implementation by adding methods for traversing the tree using **pre-order**, **in-order**, and **post-order** techniques. Finally, add a **main** method to create a binary search tree, insert multiple nodes, and invoke the traversal methods to display the tree structure. | L1, L2,L3 |
| Q2. | Construct a binary search tree from the given array of elements: {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}. Include a method called **CreateTree** to construct the binary search tree from a sorted array. This method takes an array of integers as input and constructs the tree recursively using a binary search algorithm. | L2, L3 |
| Q3. | Determine if a given binary tree is a binary search tree. You will use an **isBST** method, which takes the maximum and minimum range of the values of the nodes. | L4, L6 |
| Q4. | Design a Java program to manage country data using a **binary search tree** (BST). Create a class **Country** with members for **name** and **population**, along with a constructor and necessary methods. Define a class **BNode** to store a Country object and maintain references to its **left** and **right** children. Implement a class **BSTCountry** with a root node, a constructor, and a method to **insert** countries into the tree based on their population. Extend the BST by adding methods for **in-order** traversal, finding the country with the maximum population **(findMax)**, and finding the country with the minimum population **(findMin)**. Finally, develop a **main** method to create a BST, insert country nodes, and invoke these methods to display the results. | L5, L6 |
| Q5. | Implement a method to **remove** node **x** from a **binary search tree** while ensuring that the tree maintains its properties. The deletion process involves three cases:<br>1. **Case 1:** Node x has no children (a leaf node).<br>2. **Case 2:** Node x has one child (either left or right).<br>3. **Case 3:** Node x has two children, requiring a suitable replacement to maintain the BST structure. | L4, L6 |

| | | |
|---|---|---|
| Q6. | Write a program to implement a graph using an **adjacency matrix** and **adjacency list** representation. Develop methods to construct the graph and display its adjacency matrix and adjacency list. | L2, L3 |
| Q7. | Create a class Graph that uses a **linked list** to represent **N** vertices. Implement a constructor to initialize the graph. Add a method to read a graph and store it using an **adjacency list** representation. Additionally, implement a **Depth-First Search (DFS)** method to traverse the graph's vertices. Finally, include a **main** method to create a graph, invoke the implemented methods, and display the traversal results. | L3, L4 |
| Q8. | Implement a Java program to traverse a graph using **Breadth-First Search (BFS)** with an **adjacency list**. Use **ArrayDeque** for efficient traversal. The program should include methods to initialize the graph, add edges, display the adjacency list, and perform BFS. Finally, use the **main** method to construct the graph, invoke BFS, and display the traversal output. | L3, L4 |
| | **-END-** | |