# United International University

**Department of Computer Science and Engineering**

Course Code: CSI 217 | Course name: Data Structure and Algorithms - I

Laboratory# *Class Assignment 2*

**Total Marks: 20** **Deadline: 2 week from given date**

## Question 1 : Binary Search Tree ( Mark 10 )

**1: Insertion Operation**

Write a function **insert(root, key)** that takes a root node of a binary search tree (BST) and a value key as parameters. The function should insert the given value into the BST and return the updated root.

**2: Deletion Operation**

Implement a function **delete(root, key)** that takes a root node of a binary search tree (BST) and a value key as parameters. The function should remove the node with the given value from the BST and return the updated root.You have to handle cases where the node to be deleted has zero, one, or two children.

**3: Search Operation**

Create a function **find(root, key)** that takes a root node of a binary search tree (BST) and a value key as parameters. The function should return True if the value exists in the BST, and False otherwise.

**4: Traversal Operations**

Write functions **printPostOrder(root), printPreOrder(root), and printInOrder(root)** that take a root node of a binary search tree (BST) as a parameter and print the post-order, pre-order, and in-order traversals, respectively.

**5: Interactive Program**

Create a main function that **initializes an empty binary search tree** and then **repeatedly takes user input to perform operations**. The operations include:

- Insertion: The user provides the value to insert.
- Deletion: The user provides the value to delete.
- Search: The user provides the value to search for, and the program prints whether it exists in the tree.
- Print Traversals: The user selects the type of traversal (post-order, pre-order, or in-order), and the program prints the corresponding traversal.

**Bonus Challenge Question: Height of BST ( Mark 1)**

Write a function **height(root)** that takes a root node of a binary search tree (BST) as a parameter and returns the height of the tree (the length of the longest path from the root to a leaf node).

For each question, implement the required function and integrate it into the provided main function to create a working program. The main function should prompt the user for inputs and execute the selected operations accordingly.

**In the viva, you may be asked to explain your approach to implementing the "Delete" operation in detail. Be prepared to discuss**

## Question 2 : The Maze of Mysteries - Finding Paths with Treasures (Mark 10)

You find yourself in a mysterious maze filled with twists and turns, dead ends, and hidden treasures. Your task is to navigate through the maze to find paths from the starting position to the exit that contain hidden treasures. **The maze is represented as a 2D grid** where each cell can have the following values:

**0: Pathway.**
**1: Wall.**
**2: Exit.**
**3: Treasure.**

You have to take the maze size as input as well as the above values for that maze cell.
Write a function **find_paths_with_treasures(maze, start_row, start_col)** that takes the maze grid and your starting position as inputs. **The function should print all the paths from the**

**starting position to the exit (2) that contain treasures (3).** If no such path is possible, the function should print "No path with treasure found."

Here is a sample maze with a size of 5x5:

```
maze = [
    [0, 1, 2, 0, 0],
    [0, 0, 0, 1, 0],
    [0, 3, 1, 0, 0],
    [0, 1, 0, 1, 0],
    [0, 0, 0, 0, 0]
]
```

For the given maze and starting position (0, 0), the function should print the following path:

Path with treasure: **(0, 0)** -> (1, 0) -> (2, 0) -> **(2, 1)** -> (1, 1) -> (1, 2) -> **(0, 2)**

In this case, the path starts from the treasure at position (0, 0), follows the pathway, and ends at the exit (2) while collecting the treasure on the way.

**Your function should implement either BFS or DFS to explore the maze, avoid walls (1), and identify paths containing treasures. If no such path is found, the function should print "No path with treasure found."**

Here is a sample maze with a size of 5x5:

```
maze = [
    [0, 1, 0, 0, 2],
    [0, 0, 0, 1, 0],
    [0, 3, 1, 0, 0],
    [0, 1, 0, 1, 0],
    [0, 0, 0, 0, 0]
]
```

For this maze and starting position (1, 0), the function should print:

Path with treasure: **(1, 0)** -> (2, 0) -> **(2, 1)** -> (1, 1) -> (1, 2) ->(0,2) -> (0, 3) ->**(0,4)**
Path with treasure: **(1, 0)** -> (1, 1) -> **(2, 1)** -> (2, 0) -> (3, 0) ->(4,0) -> (4, 1) ->(4, 2) -> (4, 3) ->(4, 4) -> (3, 4) ->(2, 4) -> (1, 4) ->**(0,4)**

**During the viva, be ready to explain the logic behind your chosen approach, how you dealt with obstacles, and how you detected the exit or a treasure.**

**Bonus Challenge Question: pointing out shortest path ( Mark 1)** : implement strategy for pointing out the shortest path . In the above example 1st path is the shortest.