TASK:

Scenario: Library Book Management System

Imagine you are developing a **Library Book Management System** where each book is assigned a unique numeric **Book ID**. The system needs to efficiently manage the collection of books by supporting the following operations:

1. **Insert**: Add a new book to the library by its unique Book ID. If a book with the same ID already exists, the system should prevent duplicate entries.
2. **Search**: Allow library staff to search for a book by its Book ID to check its availability or fetch its details.
3. **Update**: Modify the details of a book (e.g., title, author, genre) if it exists in the library's collection.
4. **Delete**: Remove a book from the library's collection if it's no longer available or needed.

The library uses a **Binary Search Tree (BST)** to manage the books because it enables fast operations on the Book IDs, ensuring that the system remains efficient as the collection grows.

Implementation Logic:

1. **Nodes**:
   - Each node in the BST represents a book.
   - A node contains the following attributes:
     - book_id: The unique numeric ID of the book (used as the key for BST operations).
     - title: The title of the book.
     - author: The author's name.
     - genre: The genre of the book.
     - left and right: Pointers to the left and right child nodes.
2. **Insertion**:
   - Add a new node for the book by traversing the BST.
   - Place the new book's node in the correct position based on its book_id.
3. **Search**:
   - Traverse the BST to locate a node with the given book_id.
   - If found, display the book details; otherwise, notify that the book is not available.

4. **Update**:
   - Search for the book by its book_id.
   - If the book exists, update its attributes like title, author, or genre.
5. **Delete**:
   - Locate the node with the given book_id.
   - Remove the node using the appropriate BST deletion rules:
     - If the node is a leaf, delete it directly.
     - If the node has one child, replace it with its child.
     - If the node has two children, replace it with its in-order successor or predecessor.