Solution –Assignment-3

```
// Assignment 3: Library Book Management System
// C++ Implementation of BST and AVL Tree

#include <iostream>
#include <vector>
#include <iomanip>
using namespace std;

// Structure to represent a book
struct Book {
    int ISBN;
    double price;
    Book(int i, double p) : ISBN(i), price(p) {}
};

// Node structure for BST/AVL
struct Node {
    Book book;
    Node* left;
    Node* right;
    int height;
    Node(Book b) : book(b), left(nullptr), right(nullptr), height(1) {}
};

class BST {
protected:
    Node* root;
```

```
// Helper functions
Node* insert(Node* node, Book book) {
    if (!node) return new Node(book);
    if (book.ISBN < node->book.ISBN)
        node->left = insert(node->left, book);
    else if (book.ISBN > node->book.ISBN)
        node->right = insert(node->right, book);
    return node;
}

Node* searchByISBN(Node* node, int ISBN) {
    if (!node || node->book.ISBN == ISBN)
        return node;
    if (ISBN < node->book.ISBN)
        return searchByISBN(node->left, ISBN);
    return searchByISBN(node->right, ISBN);
}

void searchByPriceRange(Node* node, double low, double high, vector<Book>& results) {
    if (!node) return;
    if (node->book.price >= low && node->book.price <= high)
        results.push_back(node->book);
    if (node->book.price > low)
        searchByPriceRange(node->left, low, high, results);
    if (node->book.price < high)
        searchByPriceRange(node->right, low, high, results);
}
```

```
    void inorder(Node* node) {
        if (node) {
            inorder(node->left);
            cout << "ISBN: " << node->book.ISBN << ", Price: " << node->book.price << endl;
            inorder(node->right);
        }
    }

public:
    BST() : root(nullptr) {}

    void insert(Book book) { root = insert(root, book); }

    void searchByISBN(int ISBN) {
        Node* result = searchByISBN(root, ISBN);
        if (result)
            cout << "Book found - ISBN: " << result->book.ISBN << ", Price: " << result->book.price << endl;
        else
            cout << "Book not found!" << endl;
    }

    void searchByPriceRange(double low, double high) {
        vector<Book> results;
        searchByPriceRange(root, low, high, results);
        for (const auto& book : results)
            cout << "ISBN: " << book.ISBN << ", Price: " << book.price << endl;
    }

    void displayInOrder() {
```

```
        inorder(root);
    }
};

class AVL : public BST {
private:
    int getHeight(Node* node) {
        return node ? node->height : 0;
    }

    int getBalance(Node* node) {
        return node ? getHeight(node->left) - getHeight(node->right) : 0;
    }

    Node* rotateRight(Node* y) {
        Node* x = y->left;
        Node* T = x->right;
        x->right = y;
        y->left = T;
        y->height = max(getHeight(y->left), getHeight(y->right)) + 1;
        x->height = max(getHeight(x->left), getHeight(x->right)) + 1;
        return x;
    }

    Node* rotateLeft(Node* x) {
        Node* y = x->right;
        Node* T = y->left;
        y->left = x;
        x->right = T;
```

```
    x->height = max(getHeight(x->left), getHeight(x->right)) + 1;
    y->height = max(getHeight(y->left), getHeight(y->right)) + 1;
    return y;
}

Node* insert(Node* node, Book book) {
    if (!node) return new Node(book);
    if (book.price < node->book.price)
        node->left = insert(node->left, book);
    else if (book.price > node->book.price)
        node->right = insert(node->right, book);
    else
        return node;

    node->height = 1 + max(getHeight(node->left), getHeight(node->right));
    int balance = getBalance(node);

    if (balance > 1 && book.price < node->left->book.price)
        return rotateRight(node);
    if (balance < -1 && book.price > node->right->book.price)
        return rotateLeft(node);
    if (balance > 1 && book.price > node->left->book.price) {
        node->left = rotateLeft(node->left);
        return rotateRight(node);
    }
    if (balance < -1 && book.price < node->right->book.price) {
        node->right = rotateRight(node->right);
        return rotateLeft(node);
    }
```

```
        return node;
    }

public:
    void insert(Book book) { root = insert(root, book); }
};

int main() {
    AVL avlTree;

    avlTree.insert(Book(1001, 25.5));
    avlTree.insert(Book(1002, 10.0));
    avlTree.insert(Book(1003, 50.0));
    avlTree.insert(Book(1004, 20.0));
    avlTree.insert(Book(1005, 30.0));

    cout << "In-order Traversal of AVL Tree:" << endl;
    avlTree.displayInOrder();

    cout << "\nSearch for book with ISBN 1003:" << endl;
    avlTree.searchByISBN(1003);

    cout << "\nBooks in price range $15 to $35:" << endl;
    avlTree.searchByPriceRange(15, 35);

    return 0;
}
```