**CS 300 Project One Milestone Three**

**Huan Ai**

**4/6/2025**

**// 1. Course Structure**

struct Course {

std::string courseCode; // Key for BST ordering

std::string title;

std::vector<std::string> prerequisites;

};

**// 2. BST Node Structure**

struct Node {

Course course;

Node\* left;

Node\* right;

// Constructor

Node(Course c) {

course = c;

left = right = nullptr;

}

};

**// 3. BST Insert Node**

insertNode(Node\*& root, Course newCourse)

If root is null

root = new Node(newCourse)

Else If newCourse.courseCode < root->course.courseCode

insertNode(root->left, newCourse)

Else If newCourse.courseCode > root->course.courseCode

insertNode(root->right, newCourse)

Else

// Duplicate course code - handle error

End

**// 4. File Parsing**

loadCoursesIntoBST(String filePath) Returns Node\*

Create BST root = nullptr

Open input file

If file fails to open

Output "Error: File not found"

Return null

While getline(file, currentLine)

// --- Tokenization ---

vector<string> tokens

stringstream ss(currentLine)

string token

While getline(ss, token, ',')

token = trimWhitespace(token)

tokens.push\_back(token)

// --- Validation ---

If tokens.size() < 2

Output "Invalid line (missing fields): " + currentLine

Continue

// --- Create Course ---

Course newCourse

newCourse.courseCode = tokens[0]

newCourse.title = tokens[1]

// Add prerequisites (tokens 2..n)

For i from 2 to tokens.size()-1

newCourse.prerequisites.push\_back(tokens[i])

// --- BST Insertion ---

insertNode(root, newCourse)

Return root

End

**// 5. BST Search Function**

searchBST(Node\* root, String targetCode) Returns Course\*

If root is null

Return null

If targetCode == root->course.courseCode

Return &(root->course)

Else If targetCode < root->course.courseCode

Return searchBST(root->left, targetCode)

Else

Return searchBST(root->right, targetCode)

End

**// 6. Prerequisite Validation**

validatePrerequisites(Node\* root)

If root is null

Return true

// Validate current course's prerequisites

For each prereq in root->course.prerequisites

If searchBST(root, prereq) is null

Output "Missing prerequisite: " + prereq

Return false

// Recursively check left and right subtrees

Return validatePrerequisites(root->left) AND

validatePrerequisites(root->right)

End

**// 7. BST Printing Function**

printCourseInfo(Node\* root, String courseCode)

Course\* foundCourse = searchBST(root, courseCode)

If foundCourse is null

Output "Course not found"

Else

Output "Course: " + foundCourse->courseCode + " - " + foundCourse->title

If not foundCourse->prerequisites.empty()

Output "Prerequisites:"

For each prereq in foundCourse->prerequisites

Output "- " + prereq

End

**// 8. Main Program**

Main()

Initialize BST root = nullptr

// Load data

String filePath = getInput("Enter file path [default: courses.csv]: ")

If filePath is empty

filePath = "courses.csv"

root = loadCoursesIntoBST(filePath)

// Validate prerequisites

If not validatePrerequisites(root)

Exit with error

// Interactive search

While True

String input = getInput("Enter course code (or 'exit'): ")

If input == "exit"

Break

printCourseInfo(root, input)

End