// Function to open and read the course file

function loadCourseData(fileName):

// Open the file in read mode

file = open(fileName, "r")

// Initialize an empty binary search tree to store courses

courses = new BinarySearchTree<Course>()

// Initialize an empty hash table to keep track of course numbers for validation

courseNumbers = new HashTable<String, Boolean>()

// Read each line from the file

while (line = readLine(file)) is not null:

// Parse the line into tokens based on a delimiter (e.g., commas)

tokens = split(line, ",")

// Validate if the line has at least two parameters (course number and title)

if (length(tokens) < 2):

print("Error: Course information is incomplete on line:", line)

continue

// Extract course number, title, and prerequisites

courseNumber = tokens[0]

courseTitle = tokens[1]

prerequisites = extractPrerequisites(tokens[2:]) // Handle empty prerequisites as well

// Validate if prerequisites are valid courses

validPrerequisites = true

for prerequisite in prerequisites:

if (prerequisite not in courseNumbers):

print("Error: Invalid prerequisite course number:", prerequisite)

validPrerequisites = false

break

// If prerequisites are valid, create course object and add to tree

if (validPrerequisites):

course = new Course(courseNumber, courseTitle, prerequisites)

courses.insert(course)

courseNumbers[courseNumber] = true

// Close the file after reading

close(file)

return courses

// Function to extract prerequisites from the tokens (handles empty prerequisites)

function extractPrerequisites(tokens):

prerequisites = []

for token in tokens:

if (token is not empty):

prerequisites.append(token)

return prerequisites

// Course class definition

class Course:

String courseNumber

String courseTitle

List<String> prerequisites

function constructor(courseNumber, courseTitle, prerequisites):

this.courseNumber = courseNumber

this.courseTitle = courseTitle

this.prerequisites = prerequisites

// Binary Search Tree (BST) Class Definition

class BinarySearchTree:

Node root

function insert(course):

if (root is null):

root = new Node(course)

else:

insertAt(root, course)

function insertAt(node, course):

if (course.courseNumber < node.course.courseNumber):

if (node.left is null):

node.left = new Node(course)

else:

insertAt(node.left, course)

else:

if (node.right is null):

node.right = new Node(course)

else:

insertAt(node.right, course)

// Node class for BST

class Node:

Course course

Node left

Node right

function constructor(course):

this.course = course

this.left = null

this.right = null

// Function to print all course information using the binary search tree

function printCourses(courses):

// Perform an in-order traversal of the binary search tree to print courses in sorted order

inOrderTraversal(courses.root)

// In-order traversal function

function inOrderTraversal(node):

if (node is not null):

inOrderTraversal(node.left)

printCourse(node.course)

inOrderTraversal(node.right)

// Function to print course information

function printCourse(course):

print("Course Number:", course.courseNumber)

print("Course Title:", course.courseTitle)

if (course.prerequisites is not empty):

print("Prerequisites:", join(course.prerequisites, ", "))

else:

print("No prerequisites.")