// Define a Course object

CLASS Course

STRING courseNum

STRING courseName

LIST<String> prereqList

CONSTRUCTOR Course(number, name, prereqs)

courseNum = number

courseName = name

prereqList = prereqs

// Define a binary search tree to store courses

DECLARE courseTree AS BinarySearchTree<String, Course>

// Load courses from file

FUNCTION loadCourses(path)

OPEN file at path

IF file is not open THEN

PRINT "Error: Cannot open file"

RETURN

DECLARE courseLines AS empty list

FOR EACH line IN file

REMOVE whitespace from line

IF line is empty THEN

CONTINUE

ADD line TO courseLines

CLOSE file

CALL validateFile(courseLines)

CALL storeCourses(courseLines)

// Validate prerequisites exist as courses

FUNCTION validateFile(courseLines)

DECLARE courseNumbers AS empty set

// First pass: collect all course numbers

FOR EACH line IN courseLines

SPLIT line by ',' INTO tokens

IF size of tokens < 2 THEN

PRINT "Error: Each line must contain at least a course number and a title."

EXIT program

ADD tokens[0] TO courseNumbers

// Second pass: validate prerequisites

FOR EACH line IN courseLines

SPLIT line by ',' INTO tokens

FOR i FROM 2 TO size of tokens - 1

SET prerequisite = tokens[i]

IF prerequisite NOT IN courseNumbers THEN

PRINT "Error: Prerequisite " + prerequisite + " not found as a course number."

EXIT program

// Store courses in BST

FUNCTION storeCourses(courseLines)

FOR EACH line IN courseLines

SPLIT line by ',' INTO tokens

SET courseNumber = tokens[0]

SET courseTitle = tokens[1]

DECLARE prerequisites AS empty list

FOR i FROM 2 TO size of tokens - 1

ADD tokens[i] TO prerequisites

CREATE newCourse AS Course(courseNumber, courseTitle, prerequisites)

INSERT newCourse INTO courseTree WITH key = courseNumber

PRINT "Courses loaded and stored successfully"

RETURN courseTree

// Print a specific course

FUNCTION printCourse(courseTree , courseNumber)

SET course = courseTree .search(courseNumber)

IF course IS NULL THEN

PRINT "Course not found."

RETURN

PRINT course.courseNum + ": " + course.courseName

IF course.prereqList IS empty THEN

PRINT "Prerequisites: None"

ELSE

PRINT "Prerequisites:"

FOR EACH prereq IN course.prereqList

PRINT prereq

// Print all courses

FUNCTION printAllCourses(courseTree )

CALL inOrderPrint(courseTree.root)

FOR EACH course IN allCourses

PRINT course.courseNum + ": " + course.courseName

//Recursively print tree in order

FUNCTION inOrderPrint(node.left)

IF node IS NULL

RETURN

CALL inOrderPrint(node)

SET course = node.value

PRINT course.courseNum + “: “ + course.courseName

CALL inOrderPrint(node.right)

// Main menu

FUNCTION Main()

DECLARE courseTree AS new BinarySearchTree<String, Course>

WHILE TRUE

PRINT "1. Load and Validate File"

PRINT "2. Search and Print Course"

PRINT "3. Print All Courses"

PRINT "4. Exit"

PRINT "Enter choice:"

GET user input INTO choice

SWITCH choice

CASE 1:

PRINT "Enter file path: "

GET user input INTO path

SET courseTree = loadCourses(path)

CASE 2:

PRINT "Please enter course number:"

GET user input INTO courseNumber

CALL printCourse(courseTree , courseNumber)

CASE 3:

CALL printAllCourses(courseTree )

CASE 4:

PRINT "Exiting program."

RETURN

DEFAULT:

PRINT "Invalid option. Try again."