**CS 300 Project One Milestone Three**

Student Name

Date

**CS 300 Project One Milestone Three**

**Note:** *Pseudocode simplified to accommodate report length*

**Pseudocode 1: Reading and Validating Course Data from File**

| BEGIN ReadCourseData(filename)  OPEN file with filename  IF file cannot be opened THEN  PRINT "Error: File not found."  RETURN  ENDIF  INITIALIZE empty dictionary CourseDict  WHILE not end of file DO  READ line from file  SPLIT line by commas into List CourseInfo  SET courseNumber = CourseInfo[0]  SET courseTitle = CourseInfo[1]    IF length of CourseInfo < 2 THEN  PRINT "Error: Invalid format in line: " + line  CONTINUE  ENDIF  INITIALIZE empty list prerequisites  FOR each item in CourseInfo from index 2 to end DO  IF item exists in CourseDict THEN  ADD item to prerequisites  ELSE  PRINT "Error: Prerequisite " + item + " not found in course list."  ENDIF  ENDFOR    ADD courseNumber → (courseTitle, prerequisites) to CourseDict  ENDWHILE  CLOSE file  RETURN CourseDict  END ReadCourseData |
| --- |

**Pseudocode 2: Creating Course Objects and Storing in a Binary Search Tree (BST)**

| CLASS Course  PROPERTY courseNumber  PROPERTY courseTitle  PROPERTY prerequisites  FUNCTION Constructor(courseNumber, courseTitle, prerequisites)  SET this.courseNumber = courseNumber  SET this.courseTitle = courseTitle  SET this.prerequisites = prerequisites  END FUNCTION  END CLASS  CLASS Node  PROPERTY course  PROPERTY left  PROPERTY right  FUNCTION Constructor(course)  SET this.course = course  SET left = NULL  SET right = NULL  END FUNCTION  END CLASS  CLASS BST  PROPERTY root  FUNCTION Insert(course)  IF root is NULL THEN  SET root = new Node(course)  ELSE  CALL InsertRecursive(root, course)  ENDIF  END FUNCTION  FUNCTION InsertRecursive(node, course)  IF course.courseNumber < node.course.courseNumber THEN  IF node.left is NULL THEN  SET node.left = new Node(course)  ELSE  CALL InsertRecursive(node.left, course)  ENDIF  ELSE  IF node.right is NULL THEN  SET node.right = new Node(course)  ELSE  CALL InsertRecursive(node.right, course)  ENDIF  ENDIF  END FUNCTION  END CLASS  FUNCTION LoadCoursesIntoBST(CourseDict)  INITIALIZE BST tree  FOR each courseNumber in CourseDict DO  SET courseTitle = CourseDict[courseNumber][0]  SET prerequisites = CourseDict[courseNumber][1]  CREATE Course object using (courseNumber, courseTitle, prerequisites)  CALL tree.Insert(course)  ENDFOR  RETURN tree  END FUNCTION |
| --- |

**Pseudocode 3: Printing Course Information and Prerequisites from BST**

| FUNCTION PrintInOrder(node)  IF node is NOT NULL THEN  CALL PrintInOrder(node.left)  PRINT "Course Number: " + node.course.courseNumber  PRINT "Course Title: " + node.course.courseTitle  PRINT "Prerequisites: " + (", ".join(node.course.prerequisites) OR "None")  PRINT "-----------------------------"  CALL PrintInOrder(node.right)  ENDIF  END FUNCTION  FUNCTION PrintAllCourses(BST tree)  IF tree.root is NULL THEN  PRINT "No courses available."  ELSE  CALL PrintInOrder(tree.root)  ENDIF  END FUNCTION  FUNCTION PrintSpecificCourse(BST tree, courseNumber)  SET node = tree.root  WHILE node is NOT NULL DO  IF courseNumber == node.course.courseNumber THEN  PRINT "Course Number: " + node.course.courseNumber  PRINT "Course Title: " + node.course.courseTitle  PRINT "Prerequisites: " + (", ".join(node.course.prerequisites) OR "None")  RETURN  ELSEIF courseNumber < node.course.courseNumber THEN  SET node = node.left  ELSE  SET node = node.right  ENDIF  ENDWHILE  PRINT "Course not found."  END FUNCTION |
| --- |