# CS 300 Pseudocode Document

## Pseudocode is used to open files, read data from the file, parse, and check for file format errors.

The application will first try to open a CSV file and will parse it line by line if the file exists and is readable. It will validate that each line has a course number and name. Otherwise, it will throw an error and quit.  
For every 3rd, 4th, …Nth index element (if such exists) in each parsed row (end of line) and by calling the search(root)function, it will validate each prerequisite that such already exists in BinarySearchTree (BST). If a prerequisite is found in BST, create a new Course and populate it with the parsed line.

To add the new, populated course to the BinarySearchTree, the insert(course) function will be called to determine the correct location of the node in the tree.

Otherwise, if the file or line is not readable or a prerequisite does not exist in BST, Quit.

* TreeNode search(TreeNode\* node, string courseNumber)

if node == nullptr OR node->course.courseNumber == courseNumber

return the found node

if courseNumber **<** node->course.courseNumber

search(node->left, courseNumber)

if courseNumber **<** node->course.courseNumber

search(node->right, courseNumber)

* Main():

filePath = "path/to/course\_data.csv"

BinarySearchTree BST = processCourseFile(filePath);

Try to open the filename:

Parser Line = csv::Parser(filePath);

For each Line in the filePath, Try reading a line:

If line length ≥ 2:

Construct a New Course object:

Course course(lineData[0], lineData[1])

If prerequisites exist in the read line, lineData[≥2]:

Validate prerequisites:

For prerequisite in prerequisites:

Search(root, prerequisite)

If the prerequisite does not exist in BST:  
 Print "prerequisite does not exist in BST"), quit.

else

Add prerequisite to vector<string>prerequisites

course.prerequisites.push\_back(prerequisite)

ADD course to BST

insert(course) to BST.

else line length < 2:

Line format is not supported, quit.

Catch Can’tReadLineError: print("Error: Line not read at…")

Catch Can’tOpenFileError: print("Error: File not found at…")

## Pseudocode to show how to create course objects and store them in the appropriate data structure.

* Each course will be stored as a TreeNode and reside in a Binary Search Tree (BST) structure stored by its courseNumber value and initialized as follows::

class TreeNode:

Course course

TreeNode\* left

TreeNode\* right

TreeNode constructor(Course c):

course = c

left= nullptr

right = nullptr

* Each course object will be structured and initialized as follows:

class Course:

string courseNumber

string courceName

vector<string> prerequisites

Course constructor(string number = "", string title = ""):

courseNumber = number

courceName = title

* To insert each course into the BST, the insert() function will recursively determine if the new node is the tree root, or, if it is smaller or greater than the current one, and populate the new node in its appropriate branch in the BST starting from the root index going downwards.

void insert(TreeNode\* node, Course course)

if node is nullptr,

set the new course at this location

else if course.courseNumber **<** node->course.courseNumber

insert(node->left, course);

else if course.courseNumber **>** node->course.courseNumber

insert(node->right, course);

## Pseudocode that will print out course information and prerequisites.

The function printInOrder() when called with passing the root argument, will recursively traverse the Binary Search Tree (BST) from the very first parent to the smallest leaf and print each course detail by recursively jumping to a bigger node by calling printInOrder(node->right). The printCourseInfo() will print course attributes and will traverse through each prerequisite vector index if such is not empty to print the courseNumber prerequisites for each course.

void printInOrder(TreeNode\* node)

if node != nullptr

printInOrder(node->left)

printCourseInfo(node->course)

printInOrder(node->right)

void printCourseInfo(Course course)

print course.courseNumber

print course.courceName

if course.prerequisites are not empty:

for each prerequisite in the course.prerequisites vector:

Print prerequisite (courseNumber)