**ABCU course pseudocode for program that opens the file, reads the data from the file, parses each line, and checks for file format errors.:**

Function for parsing

Open the file stream using csvPath

If the file can’t be opened then

Print out an error message

Return error

Make a vector to store each line from file

Use a while loop to go through the file so while each line isn’t empty

Read line from the file

Append the line to the vector

Parse each element in the vector as the headers

Split each line in headers by commas so

For each element in the headers

Append to headers

Make a vector for the rows of the courses

Append the headers to this vector

Starting from the second elements in the original vector

Seperate each element in a line using commas (use a for loop, and split for the commas)

append them to the row vector

Iterate over the vector for rows to check for errors:

Create a vector for course numbers

Use a for loop to go over each row in the row vector,

if the length of the row is less than 2

error message is printed and an error is returned.

the first element of the row vector is appended to the vector for course numbers

Use a for loop for each row in the rows vector

if the current row isn’t matched to the current row in the vector for course numbers

print and error message and return error

Return the rows vector.

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

Function for adding course to a library

create new course object

assign course number and name from the row vector

create a new vector for the perquisites

use a for loop to add to the perquisites vector starting at 2

append the current row to the vector

assign the perquisites vector to the course

append the course to the library.

**pseudocode that will search the data structure for a specific course and print out course information and prerequisites.**

A function for searching for a course by the course’s number

use a for loop for each course in the library

if the current course number in the library matches the user’s input course number

return the current course

return an empty course if the loop finishes without finding the course

A display function for output

**ABCU course pseudocode for program that opens the file, reads the data from the file, parses each line, and checks for file format errors.:**

**Create a structure for the course information to be stored**

**Initialize the variables for the structure such as the courseid, name, the number of perquisites, and the perquisites**

**Constructor**

**class for the hash table**

**Create a bucket structure with the course, key, and pointer to next bucket**

**initialize a list for the hashtable**

**Make a function for the hash keys**

**Create an operation for hash key**

**Return**

**Function for parsing file**

**Open the csv file using the path and the libraries**

**create a temporary list for the courses**

**use a while loop for reading each row in the file**

**split the rows using commas**

**If the length of the list is 2 or more then**

**create a new course object**

**Set the course’s name and id**

**Set the prequesites**

**Use a for loop the parse through the prequesistes in the list**

**Set the prequesites and their amount in the course object**

**Add the course to the hashtable by course id**

**Add course to the list then pri**

**Create a function to initialize a list for course data**

**Function to parse the csv file**

**Initialize the path for the file**

**Load the contents of file into the list**

**Validate the information in the list**

**If the list is invalid then**

**Return error message**

**Return error**

**Function for searching**

**Create a bucket**

**find the bucket with the course ID hash**

**use a while loop to search through the buckets in the hash table**

**if the bucket is equal to the course id then return the bucket**

**go to next bucket**

Pseudocode for course structure:

CourseID, courseName, and the perquisites

TreeNode structure:

Course, left pointer, right pointer

Class for a binary search tree:

Root (with pointer)

Function for inserting:

if the root is null

the root is set to equal the new tree node

else

the function for inserting a node is called

Function for inserting a node:

if the courseID is less than the node.courseID

if the left node is null

the left node is set as the new tree node

else

the function for inserting a node is called with the left node as a parameter

else

if the right node is null

the right node is set as new Tree node

else

the function for inserting a node is called with the right node as parameter

Function for search:

called the searchnode function

Function for searchnode:

if the node is null

the course is returned

else if the courseID is < the node course ID

calls the searchnode function with left node and coursid as parameters

else

calls the searchnode function with right node and coursID as parameters

Function for printing :

If the node is not null

The print function is called for the left node as parameter

Print out the courseID, coursename, and the prerequisites

The print function is called for the right node as parameter

Function for reading files:

The file is opened using the filepath

While line is not null

Line is read

Line is split by comma and into tokens

If the length of tokens is < 2

Print error message

divide the tokens into courseID and courseName

Add the tokens into the prerequisite list using for loop

Call course class using the courseID, courseName, and prerequisites as parameters

**Pseudocode for a menu with four options:**

Create a function for menu display:

print option 1 “Load the file data into the data structure.”

print option 2 “Print an alphanumerically ordered list of all the courses in the Computer Science department.”

print option 3 “Print Course Information”

print option 4 “exit”

print for user input “your choice: “

Create a function for main:

variables for int choice

Set coursetree as binary search tree

Variable as boolean

Use a while loop:

Call the menu display function

Read the choice from the user into the int choice var

If choice is equal to 1

Call the loading course data (search tree) function

Set the boolean to true

If choice var is equal to 2

If the boolean var is false

print error message (data not loaded)

Else

call the print course list function

If choice var is equal to 3

If the booelan is false

Print error message (data not loaded)

Else

Prompt user to enter a course id

Read course id

Call the search course function with course tree and the id as parameters

Else if choice is 9

Exit the program (break)

Else

Print error message (invalid choice)

Create a function for loading the course data:

open the file at file path

if file can’t be opened:

print error message

return

while not end of file

read the line from file

parse each line into the courseid, coursename, and prerequesites

create a new course object frome this

insert into tree

close the file

Create a function for printing the course list using tree as paramteter

if the tree is empty

print error message

else

call the function for printing the tree

Create a function for searching the tree with tree and courseid as parameters

call the find course function

if the course is null

print error message

else

print the course id

print the title

print the prerequesites

Create a function for traversing

if the node is null:

return

call this function with left node

print the node courseid and node course name

call this function with right node

Create a function for finding a course

if the node is null or the node courseid is equal to courseid

return node

If courseid is less than the node.course id

return call this function to find the course id with left node as parameter

Else

Return this function with the right node as paramters

Create a function for printing the courses by vector

If the vector is empty

print error message

return

Sort the vector by course id in ascending order alphanumericly

Using for loop for each course in the vector

Print the course id and the course name

Create a function for printing the course hash table

If the hashtable is empty

print error message

return

Create an empty list

For loop for each course in hashtable

append each course to the list

Sort the list by course id in ascending order alphanumeric

For loop for each course in list

print courseid and course name

Create a function for printing course tree

If the tree is empty

print error message

return

Call the traversal function with root for course tree as parameter

Function for traversal

If the node is null

return

Call the function with left node as parameter

Print the node course id and node course name

Call the function with right node as parameter

For the vector, loading data is O(1), seach is O(n), and sort print is O(N log N), loading data and using the append method is fast but performs slowly when searching O(n).

For the hashtable loading data is O(n), search is O(1) - O(N), and sort print is O(N). The hash table could have fast look ups at O(1) if collusions can be avoided so it needs to be O(1)-O(N). Sorting is complex at O(N log N).

For the Binary tree the loading data is O(log N), search is O(log N)-O(N) and sort print is O(N). The loading is consistent at O(log N) if unbalanced then it slows to O(n) for search and insert. Insert is slower than hashtable O(log n).

The choice I will make is Binary search tree because the search O(log n) is faster than the vector, the sorting at O(n) is faster than vector sorting, and insert is equal to hash table at O(log n). Because of needing to sort Binary search tree will be more efficient at sorting than Hashtable and faster than vector.

string filename = "CS 300 ABCU\_Advising\_Program\_Input.csv"; ifstream file(filename); if (!file) { cout << "Error: Could not open file " << filename << endl; return; }