**Pseudocode for Prerequisites**

**Vector Data Structure**

Opens Files

Create function with two parameters

* Open file
* read data
* parse each line
* check for course title
* check for course number

IF the file is free of errors, check to see if prerequisite exists in course file

Function CreateCourseObj (create function’s parameters) {

* intialize variables for courses, and read file;
* open the file to read again
* WHILE file is open

store the course obj in a vector data structure

}

SearchforSpecificCourse () {

* initialize variables for opening file
* open file
* WHILE file is open
  + - * + print course information
        + store data gathered in a data structure

}

**Hash Table Pseudocode**

Start Program

* Open file
* read data
* parse each line
* check for course title
* check for course number
* IF prereq is found, add to array
  + - * IF course parameters < than two
      * - skip course
      * - display error msg: File isn't formatted Properly

- end program

ELSE

- add course name, course number, & prereq to hash table

IF prereq exists

- check to see if prereq comes before the course

- add to has table

IF prereq is not found

- skip course

- display error msg

Create function with parameters

intialize variables for courses, and read file;

open the file to read again

WHILE file is open

store the course obj in a vector data structure

SearchforSpecificCourse ()

{

initialize variables for opening file

open file

WHILE file is open

print course information

store data gathered in a data structure

}

CS-300: Milestone Tree Data Structure Pseudocode

1.

Start program

create two nodes variables for left and right;

create root variable set it to null;

create variables for course name, and an integer for course number;

open file

WHILE file is open

* read data
* parse each line

- check for course title

- check for course number

IF root != null, check if a prereq found

* + - add prereq to right node

IF course parameters are < two, add course to left node

* + - * display error msg

ELSE

* + - * add course name, num, and preq to right node
* Display Results
* close file

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

* Start Program
* create variables for course objects;
* create root variable and set it to null;
* create left and right node variables;
* Open file
* WHILE file is open
  + - * + read file
        + parse each line
        + IF root == null, SEARCH for course

IF course is found, create course object;

ADD course to right node

IF course is not found:

print error msg

close file

* stop program
* close file
* print objects
* end program

**3. Pseudocode that will print out course information and prerequisites.**

* create root variable and set it to null;
* create left and right node variables;
* open file
* WHILE file is open
  + - * + read file
        + parse each line
        + IF root is null

check for course title

check for course number

IF root == null:

check if a prereq found

add prereq to right node

IF course parameters are < two:

add course to left node

display error msg

ELSE

add course name, num, and preq to right node

display msg

close file

* + - end program

**Pseudocode for a Menu**

* create an integer for switch statement, set it to 0 name it userInput;
* Create a Bid variable to access the
* WHILE input does not equal 4;
  + - * + PRINT “1. Load Data Structure”
        + PRINT “2. Course List”
        + PRINT “3. Course”
        + PRINT “4. Exit”
        + SWITCH (integer type userInput):

case 1:

loadBids(bid);

break;

case 2:

Print: Course List;

break;

case 3:

Print Course;

break;

case 4:

Print Message: "Thank you";

end program

break;

Default:

PRINT: no input found from user;

break;

**Pseudocode that will print out the list of the courses in the Computer Science program in alphanumeric order.**

* create a sorting string with the parameter string S
* create a char that sets the length +1
* create a string to character array
* then sort the array
* create two integers for alphabet, and numbers
* While alphabet integer is less than 97:
  + - * + Increment alphabet by 1
        + IF i is less than 97, set the number int to +1
        + ELSE set the alpha int to +1
* return S
* create a string for the classes, and print the courses in alphanumeric order

5. Details of advantages and disadvantages of each data structure.

* Creating a vector is one-dimensional, which handling the data for this assignment would be a plus. Considering we are only dealing with courses, and reading files. There's also less memory that is being used when this data structure is utilized. Main disadvantage is that elements cannot be deleted, and the main one would be not being able multiple data types.
* Creating a hash table for the project allows for the information to be organized and stored. It can be called through the project with a key. This would be a huge advantage considering you are able to create, delete, and call within the project, able to create unique elements, and its's able to be synchronized. The huge disadvantage of using hash tables is that it effects the speed due to synchronization.
* Creating a tree allows for better organization, such as data can be stored in the left or right boy, which can be expanded accordingly. You are able to run searches, etc throughout the project. Similar to the hash table, it does take longer to modify.

6. Out of all three data structures, I would consider using Hash tables. Given that the disadvantage of the hash table is it can run slower, but there are many benefits. Such as being able to be more organized.

At the current status of the project, we have the basic functionality of wanting to sort courses etc, but in case that the company considered adding more functionality to the project there wouldn't be a problem adding different items to the project. With that in mind, that would be the reason for why I would consider using Hash tables so it can be updated and edited and more functions could be added to data structure.