**Project 1**

**Tyler Primas**

1. **Resubmit pseudocode from previous pseudocode assignments**

**Vector Pseudocode**

**Read file info: course number, course name, any prerequisites**

For i = 0, while i is less than the files row count, and incrementing i

CourseNum = file[i][1]

CourseName = file[i][0]

If more info is available after num and name

prereqCourse = course num in that spot

If more info is still available

prereq2 = course num in that spot

**Loop through file to create course objects and print**

currentItem = first item

While not at last item

output course number and course name

for any prerequisites

print all prerequisites until new line

output new line

currentItem = next item in list

**Search vector for specific course and print course info and prerequisites**

Specific course being searched for = user input

currentNode = head

While the current node is not pointing to a null pointer

If current node pointing to course is the same as user input

return course num, name, and prerequisites

**Hash table Pseudocode**

**Read file info: course number, course name, any prerequisites**

For i = 0, while i is less than the files row count, and incrementing i

CourseNum = file[i][1]

CourseName = file[i][0]

if more info is available after num and name

prereqCourse = course num in that spot

If more info is still available

prereq2 = course num in that spot

**Loop through file to create course objects and store them**

If key pointer is not null

CurrentList = hash table key pointer from item

Create new node

Next node = null

Node data = item

Append to end of node

**Search vector for specific course and print course info and prerequisites**

Specific course being searched for(key) = user input

currentList = hash table of item pointing to key

If current node pointing to currentList is the same as user input and not null

return course num, name, and prerequisites(data)

Else

Return null

**Tree Pseudocode**

**Read file info: course number, course name, any prerequisites**

For i = 0, while i is less than the files row count, and incrementing i

CourseNum = file[i][1]

CourseName = file[i][0]

if more info is available after num and name

prereqCourse = course num in that spot

If more info is still available

prereq2 = course num in that spot

**Loop through file to create course objects and store them**

If root pointer is null

Root = node, left pointer is null, right pointer is null

Else

Current node = root

While current node is not null

If the node is less than the current node

If current left pointer is null

Left = node

Current = null

Else

Current = left

Else

If current right pointer is null

Current right pointer = node

Current = null

Else

Current = current right pointer

Left node = null

Right node = null

**Search binary search tree for specific course and print course info and prerequisites**

Specific course being searched for(key) = user input

If current node = key

Return current node

Else if key is less than the current node

Visit left child, repeat

Else if key is greater than the current node

Visit right child, repeat

Else

Return null

Take current node that is returned and print the node pointing to the left

Print current node

Print node pointing to the right

1. **Create pseudocode for a menu**

Start program

While input is not 4:

Display: “Select a menu option”

“1. Load Data Structure”

“2. Print Course List”

“3. Print Course”

“4. Exit”

If 1 is selected:

For i = 0, while i is less than the files row count, and incrementing i

CourseNum = file[i][1]

CourseName = file[i][0]

If more info is available after num and name

prereqCourse = course num in that spot

If more info is still available

prereq2 = course num in that spot

If 2 is selected:

Starting at head node

While current node is not null

If node is greater than current node, move in front

If node is less than current node, keep behind

Continue to loop until there are no more swaps

Print out list from head to tail

If 3 is selected:

Prompt user for courseNumber

for all courses

if the course is the same as courseNumber

print out the course information

for each prerequisite of the course

print the prerequisite course information

If 4 is selected:

Print “Goodbye”

End program

1. **Design pseudocode that prints courses in alphanumeric order**

Starting at head node

While current node is not null

If node is greater than current node, move in front

If node is less than current node, keep behind

Continue to loop until there are no more swaps

Print out list from head to tail

1. **Evaluation**

For reading with any of the pseudocode, the worst-case runtime is O(N). It loops through the file to read as many times as needed providing there are lines. So, if there are 500 lines, it has to read 500 separate times. The worst case run time for creating objects for vector, hash table, and binary search tree are all O(N) because each item is created one at a time. When it comes to searching any of these, they also all have worst-case of O(N). This is because if any of the objects being searched are the very last item, they will end up with an O(N) runtime.

The advantage of a vector is that it is easy to code and easiest to understand compared to others. Vectors can insert items at O(1) so adding items is quick and efficient. The main disadvantage is that searching items is O(N) which is not great, especially if the list of items is a long one. Hash tables are much better in terms of inserting items and searching for items. On average, it will be much quicker to use than vectors. The main issue with hash tables is that they are more difficult to implement. A tree can be easier to implement than a hash table, but more difficult than vectors. Compared to the other data structures, trees are quicker at searching than vectors but slower than hash tables. So binary trees are roughly in the middle but can be quicker if implemented in the correct algorithm.

I think my plan right now is to use a binary search tree. For what the assignment is, there is not a whole lot of items in the list so it should not matter too much if it is a little bit slower than a tree. I am more comfortable with the binary search tree, and I feel like the hash table was a little harder for me to understand. If I study hash tables some more, I might be more confident with them and use that since it is quicker.