**Resubmit Pseudocode**

Vector

START

Class Course

Int numCourse

String courseName

Vector <String> preReqs

Search Course

createCourse

if (createCourse <= ifstream”FileName”)

Output “Course already exist”

Output ifStream Course information

Else

Input Course information

Break

ifStream Course

initialize Vector

initialize string line information

open ifstream

output string information from ifstream file

if (output string > 0)

Output “No file exists”

Else

Output ifstream line

Break

Data structure Course

For new course

Set count to 0

Create new course

Output Input new course information”

Input course information

Fill ifstream with course information

If (ifStream > 0)

courseNumber = token

Break

Open ifstream (sort into data structure)

Search data structure by courseNumber

If courseNumber token > 0

Output “That course Number does not exist”

Else

Output numCourse; courseName

If preReqs > 0

Output numCourse; courseName; preReqs

Output Menu Options

1. Load data structure
2. Print Course List
3. Print Course
4. Exit Program

Take input from user

1. Load data structure

Open ifstream (sort into data structure)

Break

1. Print Course List

Open dataStructure

Sort dataStructure

If (courseName = MATH)

Swap(MATH, CSCI)

Loop through list until complete

Else (CSCI = CSCI)

Std sort(numCourse)

Loop through list until complete

Output (courseName << numCourse)

Break

1. Print Course

Load dataStrcuture

Input courseName

Output “The course title: “<< courseName << endl

Output “The prerequisites for this is: “<< preReq

Break

1. Exit Program

User input – Exit

Exit Program

END

Hash Table

**START**

Open File using Fstream

If File information > 2

Read file parameters

Read each line

Parse courseNumber; Title

Output Course Information to hash table

If File information < 0

Output “File has format issues, verify information is correct”

Else

Read file parameters

Read each line

Parse courseNumber; Title; Prerequisites

Output Course Information to hash table

**Break**

Create course vector

Create Course for hash Table

Loop through file

User input int courseNumber

If input > 0

Output “You have added a Course Number”

Add Value to Hash Table

Else

Output “You didn’t add a Course Number”

User input string courseTitle

If input > 0

Output “You have added a Course Title”

Add Value to Hash Table

Else

Output “You didn’t add a Course Title”

User input string coursePrerequisites

If user input = 0

Output “You are skipping adding course Prerequisites.”

If user input > 0

Output “You had added Prerequisites”

Add Value to Hash Table

Break

Take input from courseNumber

If courseNumber != Hash Table

output “That course does not exist”

Else

Output Course number; Course title

If Prerequisites > 0

Output Course Number; Course title; Prerequisites

Break

Output Menu Options

1. Load data structure
2. Print Course List
3. Print Course
4. Exit Program

Take input from user

1. Load data structure

Open ifstream (sort into data structure (Hash Table))

Break

1. Print Course List

Open dataStructure

Sort dataStructure

If (courseName = MATH)

Swap(MATH, CSCI)

Loop through list until complete

Else (CSCI = CSCI)

Std sort(numCourse)

Loop through list until complete

Output (courseName << numCourse)

Break

1. Print Course

Load dataStrcuture

Input courseName

Output “The course title: “<< courseName << endl

Output “The prerequisites for this is: “<< preReq

Break

1. Exit Program

User input – Exit

Exit Program

**END**

Tree

**START**

iFstream to open file(“Location”)

If file < 1

Output “File does not exist, or there are format issues.”

If file > 1

Read file parameters

Read each line

Parse int courseNumber; string courseTitle; string preReqs

If parseInformation < 2

Output “There is missing course information in line”

Else

Output “adequate course information”

Store information in BST

BST = binarySearchTree

Output “File information open and stored correctly.”

**Break**

Course vector Creation

Loop through file information

Output “Please enter information type you would like to add.”

User input

If input = courseNumber

Output “Please enter courser number.”

User input int courseNumber

If courseNumber = courseNumber

Output “That course number already exist.”

Else

If input < 1

Output “Coursenumber needs to be larger than 0”

Else

Output “ courseNumber << “added successfully”

User input courseTitle

Output “Please enter course title.”

Add to BST

User input string courseTitle

If courseTitle = courseTitle

Output “That course title already exist.”

Else

If input < 3

Output “Course title needs to have a minimum of three letters.”

Else

Output “ courseTitle << “added successfully”

Add to BST

User input string preReqs

If preReqs = preReqs

Output “That preReq already exist.”

Else

If input < 3

Output “preReqs needs to have a minimum of three letters.”

Else

Output “ preReq << “added successfully”

Add to BST

**Break**

Take input from user for courseNumber

If courseNumber != BST

Output “The course information does not exist”

Else

Output courseNumber; courseTitle

If preReqs < 1

Output “There are no prereqs to show”

Else

Output courseNumber; courseTitle; preReqs

Break

Output Menu Options

1. Load data structure
2. Print Course List
3. Print Course
4. Exit Program

Take input from user

1. Load data structure

Open ifstream (sort into data structure(BST))

Break

1. Print Course List

Open dataStructure

Sort dataStructure

If (courseName = MATH)

Swap(MATH, CSCI)

Loop through list until complete

Else (CSCI = CSCI)

Std sort(numCourse)

Loop through list until complete

Output (courseName << numCourse)

Break

1. Print Course

Load dataStrcuture

Input courseName

Output “The course title: “<< courseName << endl

Output “The prerequisites for this is: “<< preReq

Break

1. Exit Program

User input – Exit

Exit Program

**END**

**Evaluation**

| **Code (Vector)** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Create Vector** | 1 | 1 | 1 |
| **Open ifstream** | 1 | n | n |
| **Sort ifstream** | 1 | n | n |
| **Create Course information.** | 1 | n | n |
| **Sort inputted course information** | 1 | n | n |
| **Append course information** | 1 | n | n |
| **Output course infromation** | 1 | n | n |
| **Total Cost** | | | 6n + 1 |
| **Runtime** | | | O(n) |
|  | | |  |
| **Code (Hash Table)** | **Line Cost** | **# Times Executes** | **Total Cost** |
| Open ifstream | 1 | n | n |
| **Read course information** | 1 | n | n |
| **Parse course information** | 1 | n | n |
| **Create vector for hashtable** | 1 | 1 | 1 |
| Input course information to hash table | 1 | n | n |
| Create course information | 1 | n | n |
| insert course information in hash table | 1 | n | n |
| Sort course information in hash table | 1 | n | n |
| **Total Cost** | | | 7n + 1 |
|  | | |  |
| **Runtime** | | | O(n) |
| **Code (Binary Search Tree)** | **Line Cost** | **# Times Executes** | **Total Cost** |
| **Open ifstream** | 1 | n | n |
| **Read information from file** | 1 | n | n |
| **Create file tree** | 1 | 1 | 1 |
| Input course information | 1 | n | n |
| **Sort course information** | 1 | n | n |
| **Create course information** | 1 | n | n |
| **Sort course information in tree** | 1 | n | n |
| **Append sorted items** | 1 | n | n |
| **Total Cost** | | | 6n + 1 |
| **Runtime** | | | O(n) |

Each sorting type has its advantages and disadvantages, and these will play different roles when implementing each one. The vector data structure has the positive of being able to grow and shrink when needed which means that the data structure can be used for multiple types of data, and there doesn’t need to be a prediction of the amount of data before creating this data structure. The downside to this is the data structure will be slower due to how it searches through the data as it must go through every piece of data before finding the results. The hash table on the other hand may be more suer friendly due to key searching, they also are space efficient which makes them a more efficient way of storing data, and they also are usually easier to use. The downside to hash tables are due to its maximum size available this means that there is only so much information that can be used in each hash table. Lastly is the binary search tree which can be quick as well, due to how it’s built it can be efficient if the information is stored correctly, they also could store information automatically when inputted. They lack on the other hand in a few ways first if the tree is built incorrectly or information is stored in a wrong fashion the tree can become unbalanced which will result in slower searches and will also increase the complexity of the tree. This means that there is more work involved with creating it and upkeeping it to maintain its searching ability and speed.

When reviewing these pros and cons and considering which one would be best used for the project, I would choose the hash table due to its pros and cons. If collisions can be kept low a hash table can be a quick and efficient sorting type. It will be easier to work with, and they can resize when information grows, they are also fast enough to keep up with the basic information such as course information, thus making it stand out from the other two sorting types.