**Psuedocode:**

**// Vector pseudocode:**  
int numPrerequisiteCourses(Vector<Course> courses, Course c) {  
   totalPrerequisites = prerequisites of course c  
   for each prerequisite p in totaldPrerequisites  
       add prerequisites of p to totalPrerequisites  
   print number of totalPrerequisites  
}

void printSampleSchedule(Vector<Course> courses) {  
   for all courses  
       print course name  
           if course has prerequisites  
               for each prerequisite  
                   print prerequisite  
}

void printCourseInformation(Vector<Course> courses, String 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  
}

**// Hashtable pseudocode**  
int numPrerequisiteCourses(Hashtable courses, Course c) {

   totalPrerequisites = Hashtable[c]  
   for each prerequisite p in totalPrerequisites  
       add prerequisites in Hashtable[p] to totalPrerequisites  
   print number of totalPrerequisites

}

void printSampleSchedule(Hashtable courses) {

   for all key, value pair in courses  
       print key course name  
           if value has prerequisites  
               for each prerequisites  
                   print prerequisites

}

void printCourseInformation(Hashtable courses, String courseNumber) {

   for all courses  
       if the course is the same as courseNumber  
           print out the course information  
           for each prerequisite of the Hashtable[course]  
               print the prerequisite course information

}

**// Tree pseudocode**  
int numPrerequisiteCourses(Tree courses, Node c) {

   totalPrerequisites = left and right child of Node c  
   for each prerequisite p in totalPrerequisites  
       add left and right Nodes of node p to totalPrerequisites  
   print number of totalPrerequisites

}

void printSampleSchedule(Tree courses) {

   for all Nodes as courses  
       print course name  
           if course has left node  
                   print left node as prerequisite  
           if course has right node  
                   print right node as prerequisite

}  
void printCourseInformation(Tree courses, String courseNumber) {

   for all Nodes  
       if the course is the same as courseNumber  
           print out the node's information  
           if course has left node  
                   print left node as prerequisite course information  
           if course has right node  
                   print right node as prerequisite course information

          end Function

       else

           if course has left node  
                   goto left node  
           if course has right node  
                   goto right node

}

-----------------------------------------------tear line----------------------------------------------------------------------------

**Runtime Analysis:**

open a file .

check for the return value of the open function .

if it is "-1":

File not found.

else:

file found

Reading the file:

the file can be read using getline .

and print using cout, or printf.

using a loop set to a condition of EOF.

-----------------------------------------------tear line----------------------------------------------------------------------------

**Advantages & Disadvantages of the three structures:**

**Vector:**

Advantages: Is an actual class, dynamic array, size can be increase, fast and one of the most popular (my choice for Project Two as well)

Disadvantages: Limited insertion, Objects exist in contiguous allocation block, require more frequent reallocation when pushing members

**Hash:**

Advantages: synchronization, efficient in terms such as table lookup for structure, widely used for that reason with software, datasets, etc.

Disadvantages: Unavoidable collision, inefficient when multiple collisions, doesn’t allow null values such as hash map

**Tree:**

Advantages: range queries, ordering of keys stored, always keep cost of insert(), delete(), lookup(), can be used in design of memory to increase speed of memory, simple algorithms, well know and often utilized

Disadvantages: can be more complicated than other times, only works if sorted and kept sorted properly, works only on element types, huge loss of efficiency since it doesn’t support random access, not good for linked list

-----------------------------------------------tear line----------------------------------------------------------------------------

**My recommendation:**

I will be utilizing vector as my data structure as that is honestly just what I am most comfortable with and what I’ve utilized the most so far in my schooling. The simplicity of it is why I am utilizing it. All of them are good options but I just have the most experience with vectors as of now. Eventually I will try to become as comfortable with Tree, and Hash as I am with vector but considering this was a project, I didn’t want to put risk it. I’m sure later in my classes I will utilize those two even more down the road. For me vector just doesn’t have many disadvantages either, it is hard to not choose it when weighing advantages and disadvantages as there is no “significant” disadvantages unlike Hash and Tree that could cause issues to arise down the road when utilizing them. So, for me it was a simple choice in which one I will use for implementation for my project.

-----------------------------------------------tear line----------------------------------------------------------------------------