**Project 1 Pseudocode**

**(Vector, Hashtable, Tree)**

**VECTOR**

// opens the course list file

**vector <Courses> openFile.courseList() {** // NA (02 Jun, 2021) geeksforgeeks.com

**create object of ifstream to open file;** // S. Samual (02-Apr-2019) tutorialspoint.com

**Vector <course>courses;**

// create an object of ifstream to store course information

**ifstream fin("abcu.txt",ios::in);**

**vector<Course> courses;**

**string line;**

// create loop to read data

// Printing course info in correct structure: cmacboyd (May 29, 2020) stackoverflow.com

**int numPrerequisiteCourses(Vector<Course> courses, Course c)** {

**totalPrerequisites = prerequisites of course c**

**for each prerequisite p in totalPrerequisites**

**add prerequisites of p to totalPrerequisites**

**print number of totalPrerequisites**

// Design a menu to load data, print course list, and to exit

**print “Menu:”**

**Print “Load Data”**

**Print “Print Course List”**

**Print “Print Course”**

**Print “Exit”**

// Using a switch case loop for menu

**Switch() {**

**Case 1: courses = load data;**

**Case 2: print course list;**

**Case 3: search courses;**

**Case 4: Exit program;**

**}**

**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**

// Printing course info in correct structure: cmacboyd (May 29, 2020) stackoverflow.com

**void printSampleSchedule(Vector<Course> courses) {**

**for all courses**

**Print out course name, courseID**

**if course has prerequisites**

**Print out list of prerequisites + courseInfo in order**

**HASHTABLE**

// creating an empty hashtable

// (10/05/2021) docs.microsoft.com

**CourseList=@{}**

// creating the item for the hashtable

**course = c**

**courseNum = n**

**prereq = p**

**courseInfo = (course, courseNum, prereq)**

// opening and reading from file

**List<string> readLines = File.ReadAllLines(ABCU.txt").ToList();**

//  [Mike Feng](https://social.msdn.microsoft.com/profile/mike%20feng/?type=forum&referrer=https://social.msdn.microsoft.com/forums/vstudio/en-US/359408e1-b302-420f-9e07-997e69d2d92f/reading-a-text-file-and-storing-it-within-a-hashtable-using-c) (Thursday, December 13) <https://social.msdn.microsoft.com/Forums/vstudio>

**System.Collections.Hashtable objTable = new System.Collections.Hashtable();  
            for course () {  
                string key = spit line;  
                string val = split line;  
                if (hashtable has key)  
                {**

**Add key();**

**}**

// Design a menu to load data, print course list, and to exit

// cmacboyd (May 29, 2020) stackoverflow.com

**print “Menu:”**

**Print “Load Data”**

**Print “Print Course List”**

**Print “Print Course”**

**Print “Exit”**

// Using a switch case loop for menu

**Switch() {**

**Case 1: courses = load data;**

**Case 2: print course list;**

**Case 3: search courses;**

**Case 4: Exit program;**

**}**

// sorting items from file: cmacboyd (May 29, 2020) stackoverflow.com

**printCourse(Course course) {**

**string courseNum = course.courseNum;**

**string courseName = course.courseName;**

**vector<string>preReq = course.preReq;**

**Print course number ;**

**Print course name ;**

**Print prerequisites;**

**for (int i = 0; i < preReq.size(); i++) {**

**print prereq list;**

**int numPrerequisiteCourses(Hashtable<Course>courses)** // NA (02 Jun, 2021) geeksforgeeks

**totalPrerequisites = c\_Hashtable**

**for each prerequisite p in totalPrerequisites**

**add prerequisites in p\_Hashtable to totalPrerequisites**

**Print preereq list**

**}**

// NA (02 Jun, 2021) geeksforgeeks

**void printSampleSchedule(Hashtable<Course> courses) {**

**for all courses**

**Print course name**

**if course has Prereq**

**for each Prereq**

**print prereq**

**}**

**BINARY SEARCH TREE**

// create the tree

// create node

// open file to read and sort the file

**Open courseList () {** // NA (02 Jun, 2021) geeksforgeeks

**void printSampleSchedule(Tree<Course> courses) {**

**if (courseNum <= current courseNum)**

**shift course left**

**}**

**else {**

**shift course right**

**print inOrder: “courseInformation: courseNum, courseName”**

**}**

**}**

// Design a menu to load data, print course list, and to exit

**print “Menu:”** // cmacboyd (May 29, 2020) stackoverflow.com

**Print “Load Data”**

**Print “Print Course List”**

**Print “Print Course”**

**Print “Exit”**

// Using a switch case loop for menu

**Switch() {**

**Case 1: courses = load data;**

**Case 2: print course list;**

**Case 3: search courses;**

**Case 4: Exit program;**

**}**

**End program**

**Evaluation**

| Code | Line Cost | # Times Executes | Total Cost |
| --- | --- | --- | --- |
| for all courses | 1 | n | n |
| if the course is the same as course number | 1 | n | n |
| print out the course information | 1 | 1 | 1 |
| for each prerequisite of the course | 1 | n | n |
| print the prerequisite course information | 1 | n | n |
| Total Cost | | | 4n + 1 |
| Runtime | | | O(n) |

**Advantages and Disadvantages**

One of the main advantages of hash tables over other data structures is the ability to add or delete elements and speed. Lookups are performed particularly fast; hash tables are very efficient when the max number of entries can be predicted in advance. The downside to using a hash table, depending on the size of the program, it can start to degrade if they go through too many collisions, as well as designing them can be very tricky. Binary Search Trees on the other hand, are a lot easier to implement than hash tables. Getting all the keys in sorted order just by doing Inorder Transversal, is not so easily done in a hash table, there are many other required steps (**H. Gupta,** 29 Mar 2022). Binary trees do have their limits though. Some disadvantages to a BTS are that they require a recursive approach which requires a lot more stack space. They also only can be used if the data is already sorted into order. Vectors are very easy to use and are able to keep track of their own size. However, resizable vectors tend to be slow. suffers from having to change and shift everything beyond the location of an insertion or a deletion.

**Recommendation**

The structure I plan to use in my project is mostly the vector structure. Vectors are useful in being able to store multiple objects, such as the courses and course numbers that’ll be using in my program. This way I can use a class to store both characters and integers for sorting my courses, and course numbers. I think my class list will be edited more easily, I can add and or delete any courses as I wish. I plan to use the push\_back() and pop\_back() functions in sorting my text file courses and prerequisites. It will consume a little more memory, but I think it will outweigh itself by being able to sort the list more effectively.

**REFERENCES**

S. Samual (02-Apr-2019) tutorialspoint.com

NA (May 29, 2020) [https://docs.microsoft.com/en-us/powershell/scripting/learn/deep- dives/everything-about-hashtable?view=powershell-7.2#hashtable-as-a-collection-of-things](https://docs.microsoft.com/en-us/powershell/scripting/learn/deep-%20%20%20%20%20%20%20%20dives/everything-about-hashtable?view=powershell-7.2#hashtable-as-a-collection-of-things)stackoverflow.com

NA (28 Feb, 2022) <https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion>

cmacboyd (Apr 17, 2014) <https://stackoverflow.com/questions/23034185/reading-back-into-a-bst-from-file>

**Himanshu Gupta (**29 Mar, 2022) https://www.geeksforgeeks.org/advantages-of-bst-over-hash-table