VECTOR PSEUDOCODE:

OPEN FILE

FOR EACH LINE IN FILE

COURSE = line.split(“,”)

COURSENUMBER = COURSE[0]

COURSETITLE = COURSE[1]

PREREQUISITES = []

IF LENGTH OF COURSE < 2

PRINT “Course data not formatted properly.”

CONTINUE

FOR i IN RANGE(2, LENGTH OF COURSE)

PREREQUISITE = COURSE[i].strip()

IF PREREQUISITE DOES NOT EQUAL “ “ AND VECTOR DOES NOT CONTAIN PREREQUISITE

PRINT “Prerequisite not found for course”, COURSENUMBER

CONTINUE

PREREQUISITES.append(PREREQUISITE)

INSERT COURSENUMBER, COURSETITLE, AND PREREQUISITES into VECTOR ALPHANUMERICALLY

IF VECTOR HAS NO COURSE NUMBER

PRINT “Course not found.”

RETURN

PRINT “Course number:”, VECTOR COURSE NUMBER

PRINT “Course title:”, VECTOR COURSE TITLE

PRINT “Prerequisites:”

FOR PREREQUISITE IN VECTOR PREREQUISITES

PREREQUISITECOURSE = VECTOR.get(PREREQUISITE)

IF PREREQUISITECOURSE EQUALS NOTHING

PRINT PREREQUISITE

ELSE

PRINT PREREQUISITECOURSE COURSE NUMBER, “,” PREREQUISITECOURSE COURSE TITLE

HASH TABLE PSEUDOCODE:

OPEN FILE

FOR EACH LINE IN FILE

COURSE = line.split(“,”)

COURSENUMBER = COURSE[0]

COURSETITLE = COURSE[1]

PREREQUISITES = []

IF LENGTH OF COURSE < 2

PRINT “Course data not formatted properly.”

CONTINUE

FOR i IN RANGE(2, LENGTH OF COURSE)

PREREQUISITE = COURSE[i].strip()

IF PREREQUISITE DOES NOT EQUAL “ “ AND HASH TABLE DOES NOT CONTAIN PREREQUISITE

PRINT “Prerequisite not found for course”, COURSENUMBER

CONTINUE

PREREQUISITES.append(PREREQUISITE)

INSERT COURSENUMBER, COURSETITLE, AND PREREQUISITES into HASH TABLE ALPHANUMERICALLY

IF HASH TABLE HAS NO COURSE NUMBER

PRINT “Course not found.”

RETURN

PRINT “Course number:”, HASH TABLE COURSE NUMBER

PRINT “Course title:”, HASH TABLE COURSE TITLE

PRINT “Prerequisites:”

FOR PREREQUISITE IN HASH TABLE PREREQUISITES

PREREQUISITECOURSE = HASH TABLE.get(PREREQUISITE)

IF PREREQUISITECOURSE EQUALS NOTHING

PRINT PREREQUISITE

ELSE

PRINT PREREQUISITECOURSE COURSE NUMBER, “,” PREREQUISITECOURSE COURSE TITLE

BINARY TREE PSEUDOCODE:

OPEN FILE

FOR EACH LINE IN FILE

COURSE = line.split(“,”)

COURSENUMBER = COURSE[0]

COURSETITLE = COURSE[1]

PREREQUISITES = []

IF LENGTH OF COURSE < 2

PRINT “Course data not formatted properly.”

CONTINUE

FOR i IN RANGE(2, LENGTH OF COURSE)

PREREQUISITE = COURSE[i].strip()

IF PREREQUISITE DOES NOT EQUAL “ “ AND BINARY TREE DOES NOT CONTAIN PREREQUISITE

PRINT “Prerequisite not found for course”, COURSENUMBER

CONTINUE

PREREQUISITES.append(PREREQUISITE)

INSERT COURSENUMBER, COURSETITLE, AND PREREQUISITES into LINKED TREE IN-ORDER

IF BINARY TREE HAS NO COURSE NUMBER

PRINT “Course not found.”

RETURN

PRINT “Course number:”, BINARY TREE COURSE NUMBER

PRINT “Course title:”, BINARY TREE COURSE TITLE

PRINT “Prerequisites:”

FOR PREREQUISITE IN BINARY TREE PREREQUISITES

PREREQUISITECOURSE = BINARY TREE.get(PREREQUISITE)

IF PREREQUISITECOURSE EQUALS NOTHING

PRINT PREREQUISITE

ELSE

PRINT PREREQUISITECOURSE COURSE NUMBER, “,” PREREQUISITECOURSE COURSE TITLE

MENU PSEUDOCODE:

IF USER INPUTS 1

PRINT “1 to load data into vector, 2 to load data into hash table, and 3 to load data into binary tree”

IF USER INPUTS 1

CALL VECTOR FUNCTION

RETURN

ELSE IF USER INPUTS 2

CALL HASH TABLE FUNCTION

RETURN

ELSE IF USER INPUTS 3

CALL BINARY TREE FUNCTION

RETURN

NOTE : our functions automatically will sort the data alphanumerically once called

IF USER INPUTS 2

IF VECTOR EQUALS NULL/EMPTY

CONTINUE

IF ELSE VECTOR IS NOT NULL/EMPTY

PRINT VECTOR

RETURN

IF HASH TABLE EQUALS NULL/EMPTY

CONTINUE

IF ELSE HASH TABLE DOES NOT EQUAL NULL/EMPTY

PRINT HASH TABLE

RETURN

IF BINARY TREE EQUALS NULL/EMPTY

CONTINUE

IF BINARY TREE DOES NOT EQUAL NULL/EMPTY

PRINT BINARY TREE

RETURN

IF USER INPUTS 3

PRINT “What class are you looking for?”

IF USER INPUT EQUALS DATA IN VECTOR, HASH TABLE, OR BINARY TREE

RETURN USER INPUT

ELSE

PRINT “Course not found.”

RETURN

IF USER INPUTS 9

PRINT “Goodbye.”

END PROGRAM

VECTOR RUNTIMES:

Operations Cost Per Line Number of Times Executed Big O Value

Opening and reading a file 1 O(n) O(n)

Parsing lines and creating objects 1 O(n) O(n)

VECTOR ADVANTAGES:

Vectors perform better when logically accessing data.

Vectors have a large amount of memory storage.

VECTOR DISADVANTAGES:

Inserting and deleting items in the middle of a vector takes a long time to execute.

Vectors have the slowest speed when it comes to searching for elements compared to hash tables and binary trees.

HASH TABLE RUNTIMES:

Operations Cost Per Line Number of Times Executed Big O Value

Opening and reading a file 1 O(n) O(n)

Parsing lines and creating objects O(1) O(n) O(n)

HASH TABLE ADVANTAGES:

Searches for elements the fastest compared to vectors and binary trees.

Retrieves elements quickly.

HASH TABLE DISADVANTAGES:

Requires a large amount of memory storage due to complexity.

BINARY TREE RUNTIMES:

Operations Cost Per Line Number of Times Executed Big O Value

Opening and reading a file 1 O(n) O(n)

Parsing lines and creating objects O(log n) O(n) O(n log n)

BINARY TREE ADVANTAGES:

Efficient in inserting and deleting elements.

Can sort data efficiently.

BINARY TREE DISADVANTAGES:

Due to how a binary tree works it uses a high amount of memory.

RECOMMENDATION:

My recommendation would be to use a hash table as they provide the fastest search times for elements which will be used in this program. Since hash tables are quick at accessing information it will be more useful compared to a vector or binary tree.