1.Resubmit

Parse\_content() returns bool:

Variable to keep track of line, and assign it to the first line of the document

While there are more lines

Check current line for first item, ensure it is correct type

Check current line for second item, ensure it is correct type

Check for third item, if there is a third item

Check first item on each line until one of them matches

If none match return false

If all three work:

Go to next line

Else return false

Struct courseObject{

Class code variable is string

name variable is string

prerequisite variable is string

prerequisite amount

}

Upload file to vector(file name):

Create a vector that hold the course object

Tell the computer to load the file name given into a file object

If Parse\_content(file) returns true continue else return

For each line in the file:

Create a new item course object

Assign the class code as the first string in the current line of the file

Assign the name as the second string in the current line of the file

If there is a third string in the current line:

Assign the prerequisite as the remaining string in the current line of the file

Else prerequisite is an empty string

Add this new course object to the vector

Print courses in vector():

For items in the vector

Print out class code for current item

Print out name of current item

If there are prerequisites

Call this function for each prerequisite

Else check next item

Upload file to hashtable(file name):

Create a vector of designated length

Tell the computer to load the file name given into a file object

If Parse\_content(file) returns true continue else return

For each line in the file:

Create a new item course object

Assign the class code as the first string in the current line of the file

Assign the name as the second string in the current line of the file

If there is a third string in the current line:

Assign the prerequisite as the remaining string in the current line of the file

Else prerequisite is an empty string

Get a hash of the class name

Check hash location in vector

If empty, put the course in the vector at this location

Else check pointer of item until it is null

Put the new course at this pointer

Print items in hashtable():

For items in vector

If item is not null

Print class code

Print class name

If item has prerequisites

Print prerequisites

Search for prerequisites in hashtable

Print pre requisites code and name

While item pointer Is not null

Print prerequisites

Search for prerequisites in hashtable

Print pre requisites code and name

Upload file to tree(file name):

Create an empty tree that will hold the course objects

Tell the computer to load the file name given into a file object

If Parse\_content(file) returns true continue else return

For each line in the file:

Create a new item course object

Assign the class code as the first string in the current line of the file

Assign the name as the second string in the current line of the file

If there is a third string in the current line:

Assign the prerequisite as the remaining string in the current line of the file

Assign prerequisite amount as amount of prerequisites

Else prerequisite is an empty string and prerequisite amount is 0

Add this new course object to the tree, by using add item function and passing in root for node

Function to add an item to the tree(node,object):

If node is empty

Make object the node

Else

If object prerequisite amount is less than node

Call this function on left node

Else

Call this function on the right node

To print out the tree(node: always pass in the root node when calling initially)

If the node passed in isn’t empty

Call this function on the left node

Print out info for this node

Call this function on the right node

2.

Menu program():

Variable choice as ineger set to 0

If not parse\_content()

Return 0

Create empty data structure

Fill empty data structure with data from parsed file

While choice doesn’t equal 9

Clear screen

Print out options to screen

Ask user for input as an integer set it equal to the choice

If user input is invalid

Choice is set equal to 9 and program ends

Match option based on choice

If choice is 1

Print course list

If choice is 2

Ask for which course they like to print

If course is valid

Print course

Else choice is set equal to 0

If choice is 9

Break

Return 0

3.

Sort then print vector(low,high):

If low is less than high

Assign p as the partition passign in the low and high

Call this quicksort function on the vector passing in low and p minus one

Call this quicksort function on the vector passing in p plus one and high

For items in vector

Print items

Sort then print hash table(): //as far as I am aware you cannot truly sort a hash table since the items end up in a hashed location

Create a variable pointer to hold the lowest item

Create a copy of the table

While items remain in copy of table

For items in copy of table

If item is smaller than varaiable

Assign item to variable

Print variable item

Delete item at pointer

Sort then print binary tree(node): // the tree is already sorted when the info is being uploaded so we just need to print from it correctly

//call this function on the root of tree first

Call this function on left

Print info for this node

Call this function on right

Evaluation:

The vector will have big o of log n since it is using quicksort. The hash table will have big o of n squared since it is essentially having to do a selection sort. The binary tree will be n because it is just traversing the tree. This means that for sorting and printing the vector will be the most time effective. I also think that for the program a vector would be best since the main functionalities fit well into a vector. A hash table would be better suited for a larger list of classes that may need to be searched often, as searching for an item is quicker in a hash table. A binary tree may be more useful if this data was going to be changing often as it is easier to insert and delete items from a tree. For this specific use a vector seems most fitting, as we know how many classes there are and they wont be changing.