­Summary

For this project, you will write a JavaScript application that, using the prerequisite structure of an academic program, responds to queries of the following types:

* 1. Given two course identifiers, is one in the pre-requisite chain of the other or not
  2. Given a course identifier, list all its pre-requisite chains
  3. Given a JSON file that contains the prerequisite definition of an academic department, print it in a simple form

Files used in this project

In the description of this project, we will reference the following two files.

* 1. [The prerequisite requirements for the CS courses (CS-BS\_course\_requisites\_v2\_pp.json)](https://canvas.sonoma.edu/courses/40218/files/3849249?wrap=1)[Download The prerequisite requirements for the CS courses (CS-BS\_course\_requisites\_v2\_pp.json)](https://canvas.sonoma.edu/courses/40218/files/3849249/download?download_frd=1). This file contains a JSON Object. One of its keys is called "courses" and it is the value of this key that we will use in this project. The value of "courses" itself is an object and its keys are the course IDs that we will use when we query this particular JSON file. The course IDs in this file are mostly those of the CS courses. The MATH course IDs in this file are those that are potential prerequisites of some CS course.
  2. [The prerequisite requirements for the MATH courses (MATH-BS\_course\_requisites\_pp.json).](https://canvas.sonoma.edu/courses/40218/files/3849235?wrap=1)[Download The prerequisite requirements for the MATH courses (MATH-BS\_course\_requisites\_pp.json).](https://canvas.sonoma.edu/courses/40218/files/3849235/download?download_frd=1)The structure (not its content) of tis JSON file is identical to the previous JSON file. The course IDs in this file are mostly those of the MATH courses.

A Few Definitions

A prerequisite chain is a sequence of courses that must be taken prior to taking a given course. We are only interested in *maximal* prerequisite chains. For example, consider CS 415. Even though any of the sequence, CS 415, CS 315; CS 415, CS 315, CS 215; and CS 415, CS 315, CS 215, CS 115 would be a prerequisite chain for that course, for this project, we only take the last one into consideration as it describes a node-to-leaf path in the prerequisite graph (see below).

Some courses have more than one prerequisite chain. For example, the following is a list of (maximal) prerequisite chains for CS 315. We will learn more about how to generate these chains later in this write up.

CS 315, CS 215, CS 115  
CS 315, CS 210, CS 115  
CS 315, CS 242, CS 115  
CS 315, CS 242, MATH 161

As another example, the prerequisite chains for CS 252 is listed below.

CS 252, CS 215, CS 115  
CS 252, CS 242, CS 115  
CS 252, CS 242, MATH 161

When a course has multiple prerequisite chains, they may be listed in any order, but they must be distinct (no duplicates).

One of our goals for this project is to determine whether two courses *can be taken concurrently* (during the same semester), given the prerequisite structure of the courses of an academic department. For example, we may want to know whether or not CS 315 and CS 470 be taken by some student during the same semester while honoring the prerequisite requirements of each course.

Obviously, if one course is the prerequisite of the other course, as it is the case with CS 315 and CS 470, then they cannot be taken concurrently by a student during the same semester. What if neither of the two courses is a direct prerequisite of the other one? How do we determine if they can be taken concurrently or not? For example, can CS 215 and CS 479 be taken concurrently?

Two courses can be taken concurrently if neither one is in any of the prerequisite chains of the other. The prerequisite chains of CS 479 are:

CS 479, CS 315, CS 215, CS 115   
CS 479, CS 315, CS 210, CS 115   
CS 479, CS 315, CS 242, CS 115   
CS 479, CS 315, CS 242, MATH 161

CS 215 has only one prerequisite chain and it is:

CS 215, CS 115

CS 479 is not on the prerequisite chain of CS 215. However, CS 215 is on one of the prerequisite chains of CS 479. Therefore, CS 215 and CS 479 cannot be taken concurrently. To take CS 479, the students have to have taken CS 215 in some prior semester.

On the other hand, CS 315 and CS 351 can be taken concurrently. This is because CS 315 is not on any of the prerequisite chains of CS 351 and vice versa. The prerequisite chains of these two course have been listed below.

The prerequisite chains of CS 315 are:

CS 315, CS 215, CS 115   
CS 315, CS 210, CS 115   
CS 315, CS 242, CS 115   
CS 315, CS 242, MATH 161

The prerequisite chains of CS 351 are:

CS 351, CS 215, CS 115   
CS 351, CS 252, CS 215, CS 115   
CS 351, CS 252, CS 242, CS 115   
CS 351, CS 252, CS 242, MATH 161

Context

We will use the JSON file that contains CS course IDs (CS-BS\_course\_requisites\_v2\_pp.json) to discuss this project (as mentioned above, the structure of MATH-BS\_course\_requisites\_pp.json is the same.) Additionally, any reference that we make will be a reference to the contents of the Object whose key in the JSON file is labeled "courses".

Using "courses" Object, let's look at a few examples in detail.

CS 115 doesn't have a prerequisite chain (the length of its prerequisite chain is zero) because of the following key/value pair.

 "CS 115": {  
            "min\_grade": "C-",  
            "catalog": "115",  
            "course\_pre\_reqs": [],  
            "subject": "CS"  
},

This is due to the fact that the value of *course\_pre\_reqs* is empty.

On the other hand, the prerequisite chain of CS 215 is:

CS 215, CS 115

This is due to key/value pair of CS 215, which has been copied below.

"CS 215": {  
            "min\_grade": "C-",  
            "catalog": "215",  
            "course\_pre\_reqs": [  
                {  
                    "courses": [  
                        {  
                            "CS 115": {  
                                "min\_grade": "C-",  
                                "catalog": "115",  
                                "subject": "CS"  
                            }  
                        }  
                    ],  
                    "num\_required": 1  
                }  
            ],  
  
            "course\_co\_reqs": [  
                {  
                    "courses": [  
                        {  
                            "CS 210": {  
                                "catalog": "210",  
                                "subject": "CS"  
                            }  
                        }  
                    ],  
                    "num\_required": 1  
                }  
            ],  
            "subject": "CS"  
        },

course\_pre\_reqs.courses array has one element:

{  
      "CS 115": {  
              "min\_grade": "C-",  
              "subject": "CS",  
             "catalog": "115"  
      }  
}

The key of that element, as it is the case with all elements that describe pre-requisites, is one of (in this case, the only) prerequisite courses for CS 215. After having discovered that CS 115 is one of CS 215's prerequisites, we look up its definition in the JSON file. You have already seen the prerequisite definition of CS 115. It is empty. Therefore, CS 115 is the only other item in the chain of prerequisites of CS 215. CS 215 has only one chain because, CS 115 had only one chain (an empty chain) and CS 215 contains only one item, CS 115, in its course\_pre\_reqs array.

Please note that for this project, we are ignoring co-requisites.

On the other hand, CS 315 and CS 450 can not be taken concurrently because CS 315 is on at least one of the prerequisite chain of CS 450. It is in fact on 5 of the prerequisite chains of CS 450. Here is the prerequisite chains for CS 450.

CS 450 CS 252 CS 215 CS 115   
CS 450 CS 252 CS 242 CS 115   
CS 450 CS 252 CS 242 MATH 161   
CS 450 CS 252 CS 242 MATH 161X   
CS 450 CS 315 CS 210 CS 115   
CS 450 CS 315 CS 215 CS 115   
CS 450 CS 315 CS 242 CS 115   
CS 450 CS 315 CS 242 MATH 161   
CS 450 CS 315 CS 242 MATH 161X   
CS 450 CS 315 MATH 142 MATH 161   
CS 450 CS 315 MATH 142 MATH 161X   
CS 450 CS 315 MATH 142 MATH 161A   
CS 450 CS 315 MATH 142 MATH 161B

The Project

For this project, you will write a JavaScript program that responds to the following operations.

* 1. Prints a simplified version of the underlying pre-requisite JSON file
  2. Given two course IDs, determines if they can be taken concurrently or not
  3. Given a course ID, it prints all of its pre-requisite chains

Each command starts with one of the following two options.

* 1. -cs -- this option means you should use the CS prerequisite JSON file to run the command
  2. -math -- this option means you should use the MATH prerequisite JSON file to run the command

The exact command-line options and their corresponding output will follow. After each command, your program should stop. In other words, your solution responds to one command at a time. Follow the expected output closely and do not print anything that is not part of the expected output. If you have not implemented an option, your output for option should be: "option-name has not been implemented." For example, "-simplify has not been implemented."

***-simplify***

The command line for this option follows:

node project1.js -cs -simplify

Using the prerequisite JSON file for CS, print its simplified pre-requisite structure. The output should look like the following. It has been generated by the use of this JavaScript statement: console.log(preReqSimplified, null, '\t'); -- preReqSimplified is a JavaScript Object that has been created and populated with the key/value pairs for each key in the "courses" (see above for the definition of this key)

{  
  'CS 115': [],  
  'CS 210': [ 'CS 115' ],  
  'CS 215': [ 'CS 115' ],  
  'CS 242': [ 'CS 115', 'MATH 161', 'MATH 161X' ],  
  'CS 252': [ 'CS 215', 'CS 242' ],  
  'CS 315': [ 'CS 210', 'CS 215', 'CS 242', 'MATH 142' ],  
  'CS 330': [ 'CS 315' ],  
  'CS 340': [ 'CS 215', 'CS 252' ],  
  'CS 349': [ 'CS 315' ],  
  'CS 351': [ 'CS 215', 'CS 252' ],  
  'CS 355': [ 'CS 215' ],  
  'CS 360': [ 'CS 315' ],  
  'CS 365': [ 'CS 210', 'CS 215', 'CS 252' ],  
  'CS 370': [ 'CS 215' ],  
  'CS 375': [ 'CS 215', 'MATH 161' ],  
  'CS 385': [ 'CS 370' ],  
  'CS 386': [ 'CS 315', 'CS 355' ],  
  'CS 391': [ 'CS 215' ],  
  'CS 415': [ 'CS 315' ],  
  'CS 425': [ 'CS 315' ],  
  'CS 450': [ 'CS 252', 'CS 315' ],  
  'CS 454': [ 'CS 315' ],  
  'CS 460': [ 'CS 252', 'CS 315' ],  
  'CS 470': [ 'CS 315', 'CS 370' ],  
  'CS 479': [ 'CS 315' ],  
  'CS 480': [ 'CS 315' ],  
  'MATH 142': [ 'MATH 161', 'MATH 161X', 'MATH 161A', 'MATH 161B' ],  
  'MATH 161': [],  
  'MATH 161X': [],  
  'MATH 161A': [],  
  'MATH 161B': []  
}

***-together***

The entire command line for this option follows:

node project1.js -cs -together "Course ID1" "Course ID2"

Can the courses specified by Course ID1 and Course ID2 be taken concurrently? The answer is one of the following:

Course ID1 and Course ID2 can be taken concurrently.  
Course ID1 and Course ID2 cannot be taken concurrently.

For example:

node project1.js -cs -together "CS 215" "CS 454"

CS 215 and CS 454 cannot be taken concurrently.

Another one:

node project1.js -cs -together "CS 450" "CS 460"

CS 450 and CS 460 can be taken concurrently.

***-chains***

The entire command line for this option follows:

node project1.js -cs -chains "Course ID"

Print the list of prerequisite chain(s) for the course specified by Course ID. Each chain should get printed on one line. The order in which the chains get printed is not important. Here is an example:

node project1.js -math -chains "MATH 220"

The corresponding output (in any order):

MATH 220 MATH 161   
MATH 220 MATH 161B MATH 161A   
MATH 220 MATH 161X   
MATH 220 MATH 250 MATH 161   
MATH 220 MATH 250 MATH 161B MATH 161A   
MATH 220 MATH 250 MATH 161X   
MATH 220 MATH 250 MATH 300B MATH 300A   
MATH 220 MATH 255   
MATH 220 MATH 265 MATH 161   
MATH 220 MATH 265 MATH 161B MATH 161A   
MATH 220 MATH 265 MATH 161X   
MATH 220 MATH 265 MATH 250 MATH 161   
MATH 220 MATH 265 MATH 250 MATH 161B MATH 161A   
MATH 220 MATH 265 MATH 250 MATH 161X   
MATH 220 MATH 265 MATH 250 MATH 300B MATH 300A   
MATH 220 MATH 381 MATH 265 MATH 161   
MATH 220 MATH 381 MATH 265 MATH 161B MATH 161A   
MATH 220 MATH 381 MATH 265 MATH 161X   
MATH 220 MATH 381 MATH 265 MATH 250 MATH 161   
MATH 220 MATH 381 MATH 265 MATH 250 MATH 161B MATH 161A   
MATH 220 MATH 381 MATH 265 MATH 250 MATH 161X   
MATH 220 MATH 381 MATH 265 MATH 250 MATH 300B MATH 300A   
MATH 220 MATH 445 MATH 345 MATH 261 MATH 211 MATH 161   
MATH 220 MATH 445 MATH 345 MATH 261 MATH 211 MATH 161B MATH 161A   
MATH 220 MATH 445 MATH 345 MATH 261 MATH 211 MATH 161X   
MATH 220 MATH 445 MATH 345 MATH 261 MATH 241 MATH 211 MATH 161   
MATH 220 MATH 445 MATH 345 MATH 261 MATH 241 MATH 211 MATH 161B MATH 161A   
MATH 220 MATH 445 MATH 345 MATH 261 MATH 241 MATH 211 MATH 161X

Support files

[Here is a zip file](https://canvas.sonoma.edu/courses/40218/files/3849951?wrap=1)[Download Here is a zip file](https://canvas.sonoma.edu/courses/40218/files/3849951/download?download_frd=1) a WebStorm project (please see course syllabus about this application) that contains an initial JavaScript for this project. It contains the code for check the command line arguments based on the requirements of this argument. We have provided this in case you are not familiar with handling command line arguments. You do not have to use WebStrom to run it. You can run it like this on blue.cs. The command is run in project1, a directory that gets created when you unzip this file.

unzip project1.zip  
cd project1  
node -cs -together "CS 315" "CS 450"

What to turn in

This is a small project and as such, you can write its solution in a single JavaScript file. Of course, that file will have to import (require) the prerequisite JSON files as they have been provided to you.

How we test your submission

Suppose that the name of the script that you submit is project1.js. As an example, to test it for -togther options, we will do:

node project1.js -cs -together "CS 450" "CS 215"

This is consistent with the requirements of this option. Your output should be in format exactly like the samples that we have provides.  Please do not put yourself in the position of losing points by not following the directions closely. It is as important as a correct solution to this project.