**Prerequisite String Formatting:**

If a class has no prerequisites, the string should be empty or null.

When a class has a prerequisite, separate the course letters and course number with an *underscore*. This would make comparing classes easier and could be easily changed on the front end if needed.

Graphical user interface, text, application, email

Description automatically generatedString: **APRL\_225**

When there is a list of classes that all must be taken, place a *comma* between the courses. A comma serves as an “AND” operator.  
Graphical user interface, text, application

Description automatically generated

String: **APRL\_373,APRL\_390,APRL\_464,APRL\_394**

When there is an “or” option between two courses, place a *vertical bar* between the courses.

**Timeline

Description automatically generated**

String: **APRL\_373|BURTL\_395**

Here’s another example:

Text, application

Description automatically generated with medium confidence

String: **MATH\_151|MATH\_153|MATH\_156**

Nearly all courses have prerequisites that can be covered simply using commas and vertical bars. The next page has a few examples combining AND and OR together.

Text

Description automatically generated

String: **MATH\_180|MATH\_270,MATH\_275|MATH\_250,CS\_145**

Graphical user interface, text, application

Description automatically generated

String: **CS\_145,MATH\_158,MATH\_275,STAT\_330|STAT\_331**

Graphical user interface, text, application

Description automatically generated

String: **MATH\_180|MATH\_270,MATH\_250|MATH\_275**

There sometimes exists situations where there are multiple prerequisite paths that may be taken. In this rare case, use an *ampersand* to indicate a sort of embedded AND operation. Here is an example:

Graphical user interface, text, application

Description automatically generated

String: **GDD\_325,CS\_326&CS\_358|DES\_350**

For co-enrollment prerequisites, preface a class with an *exclamation point*.

Text, application

Description automatically generated

String: **!BIO\_332|CHEM\_311|CS\_244**

There also exist rare cases where there is a minimum class or higher that could be taken. If you encounter this, place a *greater than sign* at the front of the class. If Xs are present instead of a specific course number, replace the Xs with 0s.

Application

Description automatically generated with low confidence

String: **NANO\_230|>CHEM\_200|>PHYS\_281**

**Parsing Algorithm:**

Quick notes:

1. In general, the hierarchy of parsing would be as follows:

* , “AND” is evaluated first
* | “OR” is evaluated next
* & “embedded-AND” is evaluated last

1. There exist several nice string parsing methods in js that we could use:

<https://www.w3schools.com/jsref/jsref_includes.asp>  
<https://www.w3schools.com/jsref/jsref_split.asp>

Includes() could be used to detect specific characters in the string and split() could be used to separate one string into an array of multiple strings.

1. Any time I mention “prerequisite check”, I mean an algorithm is ran to see if the course appears prior to the currently selected semester on the course planner. (or in the same semester in the case of co-enrollment prerequisites)

When parsing through a string, this is how I envision it will be processed:

1. First split the string into multiple strings via the comma AND.
2. If a split string is only one course, we run the prerequisite check, else we enter **step 3**. If the prerequisite check is satisfied (returns true because the prerequisite was found), then we discard the string because it is no longer needed. If false is returned, then we notify the user that the prerequisite is not valid and return the string that did not satisfy the needed prerequisite.
3. We check the string and if it contains a vertical bar, we go to **step 4**. Else go to **step 5.**
4. Entering this step means the string has a vertical bar, so we split the string into multiple strings via the vertical bar. For each of the resulting strings that are one course, we run the prerequisite check. If a string is not one course, we go to **step 5** to further split the string. If the prerequisite check returns true for any of the split strings, we can discard the entire unsplit string and any “child” strings. If all the strings ran the prerequisite check and all returned false, then we return the unsplit string to notify the user of the invalid prerequisites.
5. Entering this step means the string has an ampersand, so we split the string into multiple strings via the ampersand. From here, all strings are a specific course, so we run the prerequisite check on each string. If any return false, we return the unsplit string to notify the user of invalid prerequisites. Otherwise, (all returned true) we discard the unsplit string and any “child” strings.
6. If there are no strings or child strings remaining, the prerequisites are satisfied.

Here is an example of the process in action:

Original String (for class GDD\_450): **GDD\_325,CS\_326&CS\_358|DES\_350**

First, we evaluate the comma AND, which gives us two strings:

String1: **GDD\_325** String2: **CS\_326&CS\_358|DES\_350**

Since String1 has only one course (it does not contain & or |), so we run the prerequisite check. Here we can go down two paths:

1. If the prerequisite check for String1 returns **false**, we do not need to process any further, and we return String1 to indicate that that course failed the prerequisite test.
2. If the prerequisite check for String1 returns **true**, then can then discard String1 and examine String2. String2 is not one course, so we now split by OR to get the following strings:

String2: **CS\_326&CS\_358|DES\_350**String2.1: **CS\_326&CS\_358** String2.2: **DES\_350**

Since String2.2 has only one course, we run the prerequisite check. Here we can go down two paths:

1. If the prerequisite check for String2.2 prerequisite returns **true**, we discard the unsplitstring and all “child” strings. We are then left with no strings, which means that all prerequisites are satisfied.
2. If the prerequisite check for String2.2 prerequisite returns **false**, we need to examine all other strings, which in this case is only String2.1. Since String2.1 is not one single class, we need to split it via the ampersand, resulting in the following strings:

String2: **CS\_326&CS\_358|DES\_350**String2.1: **CS\_326&CS\_358** String2.2: **DES\_350** String2.3: **CS\_326** String2.4: **CS\_358**

We now run the prerequisite check on the resulting strings (String2.3 and String2.4). Here are the possible scenarios:

1. CS\_326 and CS\_358 are false. -> false (does not meet necessary prerequisites)
2. CS\_326 is false, CS\_358 is true -> false
3. CS\_326 is true, CS\_358 is false -> false
4. CS\_326 and CS\_358 are true -> true

Only for result four (both are true) would we delete the unsplit string. We would then be left with no strings, so the prerequisites are met. For results one through three, we would notify the user that the prerequisites are not met and return the unsplit string as the reason.

Here’s another example:

Original String (for class MSCS\_396): **MATH\_180|MATH\_270,MATH\_275|MATH\_250,CS\_145**

First, we evaluate the comma AND, which gives us three strings:

String1: **MATH\_180|MATH\_270**

String2: **MATH\_275|MATH\_250**

String3: **CS\_145**

Since String3 is one course, we can run the prerequisite check. Here we can go down two paths:

1. If the prerequisite check for String3 returns **false**, we do not need to process any further, and we return String3 to indicate that that course failed the prerequisite test.
2. If the prerequisite check for String3 returns **true**, then can then discard String3 and examine String1. String1 is not one course, so we now split by OR to get the following strings:

String1: **MATH\_180|MATH\_270**

String1.1: **MATH\_180**

String1.2: **MATH\_270**

String2: **MATH\_275|MATH\_250**

We then run the prerequisite check for String1.1. We again can go down two paths:

1. If the prerequisite check for String1.1 returns **true**, we do not need to process any further. We delete String1 and all child strings. We then examine String2 (not elaborated on here).
2. If the prerequisite check for String1.1 returns **false**, we run the prerequisite check for all other child strings. So, we would then run the prerequisite check for String1.2. We can go down two paths here:
   * + 1. If the prerequisite check for String1.2 returns **true**, we do not need to process any further. We delete String1.2 and all child strings. We then examine String2 (not elaborated on here).
       2. If the prerequisite check for String1.2 returns **false**, we run the prerequisite check for all other child strings; however, there are no other child strings. This means that we failed the prerequisites and return the unsplit string as the source of the where the prerequisites fail.