# Semantic Versioning Problem

A semantic version (semver) number or semver is a version number used to label different versions of software to provide developers with information on the impact of upgrading to a new version.

**Your mission, should you choose to accept it, is as follows**:
In one of Java, Python or JavaScript, write a function called `determinePrecedence` that takes two valid semantic versions and returns a boolean. The boolean should be true if the first semver provided to the function has a higher precedence than the second semver provided, according to the rules of precedence.

The semvers will be provided to the function as strings i.e. “0.1.2” or “3.4.0-alpha”.

For the purposes of this problem, you can assume that all versions provided are in a valid format, and build metadata has already been removed from the string.

When you have finished your function, put the code for your solution on github and send us a link to your repository. Your repository should contain instructions on how to execute the function and any tests that you may have provided.

Rules for semantic versioning can be found at https://semver.org/, however we’ve highlighted a few useful ones below.

**Determining Semantic Version Precedence:**
Precedence refers to how versions are compared to each other when ordered.

1. Precedence is determined by the first difference when comparing each of the identifiers from left to right as follows: Major, minor, and patch versions are always compared numerically.
   Example: `1.0.0` < `2.0.0` < `2.1.0` < `2.1.1`.
2. When major, minor, and patch are equal, a pre-release version has lower precedence than a normal version:
   Example: `1.0.0-alpha` < `1.0.0`.
3. Precedence for two pre-release versions with the same major, minor, and patch version MUST be determined by comparing each dot separated identifier from left to right until a difference is found as follows:
   1. Identifiers consisting of only digits are compared numerically.
   2. Identifiers with letters or hyphens are compared lexically in ASCII sort order.
   3. Numeric identifiers always have lower precedence than non-numeric identifiers.
   4. A larger set of pre-release fields has a higher precedence than a smaller set, if all of the preceding identifiers are equal.

**Semantic Version Format**:

Semvers take the form X.Y.Z where X, Y, and Z are non-negative integers, and MUST NOT contain leading zeroes. X is the major version, Y is the minor version, and Z is the patch version.

- `0.0.1, 1.11.2` and `2.3.0` are all **valid** semvers.
- `09.1.0`, `10.01.9` and `1.2.04` are **invalid** because they contain leading zeros

A pre-release version MAY be denoted by appending a hyphen and a series of dot separated identifiers immediately following the patch version. Identifiers MUST comprise only ASCII alphanumerics and hyphens [0-9A-Za-z-]. Identifiers MUST NOT be empty. Numeric identifiers MUST NOT include leading zeroes.

- `1.0.0-alpha, 1.0.0-alpha.1, 1.0.0-0.3.7, 1.0.0-x.7.z.92`, and `1.0.0-x-y-z.-` are all **valid** semvers
- `1.0.0-2/8, 1.0.0-~` and `1.0.0-1o*` are all **invalid** semvers because they use non-alphanumeric characters
- `1.0.0-.-, 1.0.0-1..2`, and `1.0.0-1.` are all **invalid** semvers because they have empty identifiers
- `1.2.3-09` is an **invalid** semver because it contains a leading 0