I used C# using the .NET Core 2.0 Framework to implement this solution.

I implemented a static class called ManifestParser (ManifestParser.cs) that reads the manifest files and parses out the data to hydrate a collection of Manifest Records. The static method ParseManifestFile is responsible for reading in each line of the file and parsing out the contents to hydrate a ManifestRecord object. The ParseManifestFile method returns a collection of Manifest records that I used as an in-memory cache throughout the application to retrieve the stadium manifest data. Please note that I could have also used a dictionary where the sectionId would have been my key and a list of ManifestRecord objects for that corresponding section Id would have been my value.

The ManifestRecord (ManifestRecords.cs) is a model object that represents the information within our domain such as the section Id, section name, rowed, row and name. I added a property called IsSuite to represent the scenario where some sections are not subdivided into rows such as suites.

My implementation of the readManifest() method on the Normalizer class invokes the method ParseManifest on the static ManifestParser class. I also added a private member field called \_ManifestRecords an in-memory cache to access the stadium manifest.

My implementation of the Normalizer method attempted to match the provided section description with the section name from the manifest utilizing string comparison. I implemented a method called LookUpSectionName that takes in the provided section description and attempts to parse it out to a corresponding section name in the stadium manifest based on the numeric digits in the provided section description.

I then proceeded to lookup the section id based on a direct string comparison between the section description in the manifest with the return value from the LookUpSectionName method. I also added another comparison check utilizing the string contains that checks for a specific partition in the manifest section description with the provided description. If this comparision was utilized (i.e. bContainsComparision was set to true) an additional step was utilized in an attempt to find the section Id by utilizing a regular expression and also incorporating the provided row id in an attempt to better filter the results.

If there were no matches, I invalided the section input. If there was a corresponding match, I would then attempt to retrieve the row id.