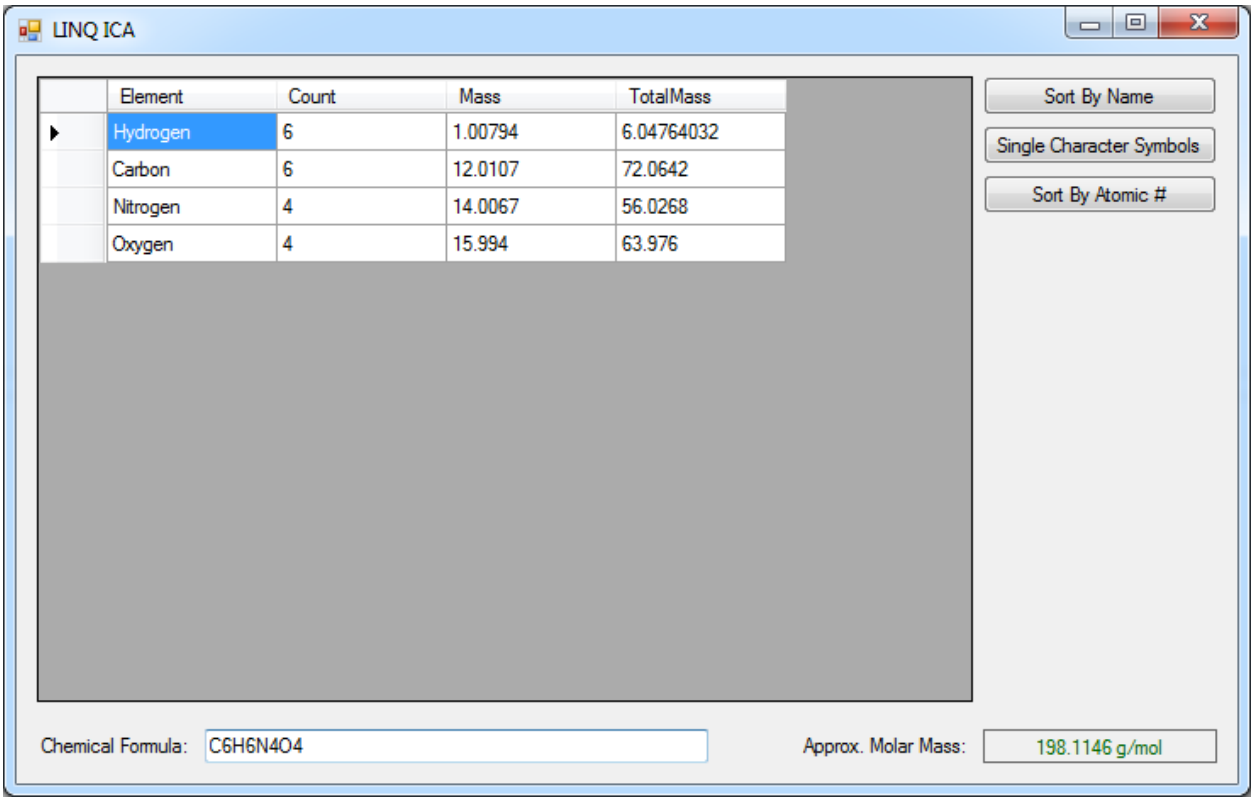
CMPE2800 – Molecular Weight Calculator (LINQ Assignment)

This assignment will be treated as a real world project, as in there is no solution prepared ahead of time – you are building it. Your team will need to self-coordinate the activities of the group. Groups and SVN repos will be generated ahead of time.

The assignment is to create a molar mass calculator. Given a chemical formula, the molar mass calculator will produce the sum of the atomic weights for each atom in the formula times its occurrence.

The screenshot below is a proof of concept example of what the output would look like:

Given : 2,4-Dinitrophenylhydrazine ($C_6H_6N_4O_4$)

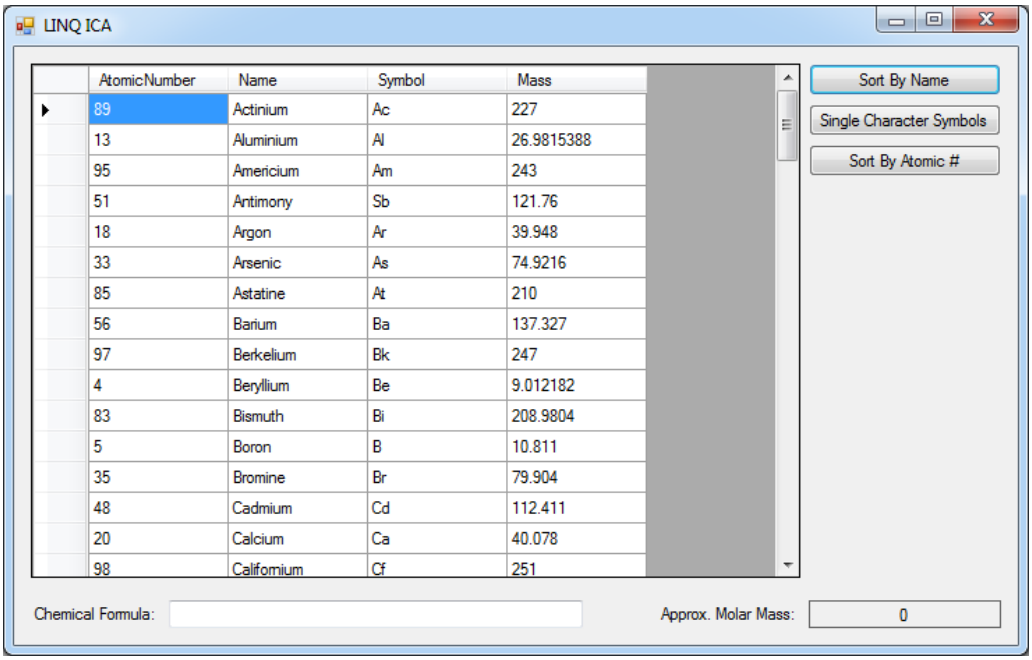


This assignment will involve a number of technologies, and some research and grunt work will be required to complete it. You will be given nothing to start with in this assignment, so all data will need to be collected (for example, components of the periodic table) and integrated into the product.

While you are free to implement much of this assignment any way you see fit, specific technologies that will be of help are regular expressions, and the use of a **`DataGridView`** (the main control in the sample above). Your instructor will discuss the use of this control with the anonymous types produced by LINQ expressions.

Where possible, you must manage and display information using LINQ.

The nerds paying you for this application insist on having the three top right buttons. Each will display the full periodic table of elements as specified. For example, when sorted by name, the output would be:



These buttons will clear the input and override all current output.

You are only required to manage elements with atomic number 1-99 inclusive, but you may do more.

As the user types in the chemical formula, you will attempt to actively parse the elements referenced and their counts. It is recommended that you use the Regex class to do this.

You will use color coding in the molar mass display to indicate success/failure of the parsing process.

It is recommended that you use a **`Dictionary`** to hold the symbol for the element and its count as you parse. There are two reasons for this:

- The user is allowed to specify elements multiple times (e.g.: AlOHOHOH (aluminum hydroxide))
- You **must** use a join in LINQ to lookup the formula against the table of elements (key == Symbol)

All results shown in the **`DataGridView`** must come from anonymous types produced by LINQ expressions.

You will be required to get all other specifications for this application from your instructor.

Assuming it still exists at the time you are doing this, the site: https://regex101.com/ may be very helpful in building and testing your regular expressions. Check for Moodle links too.