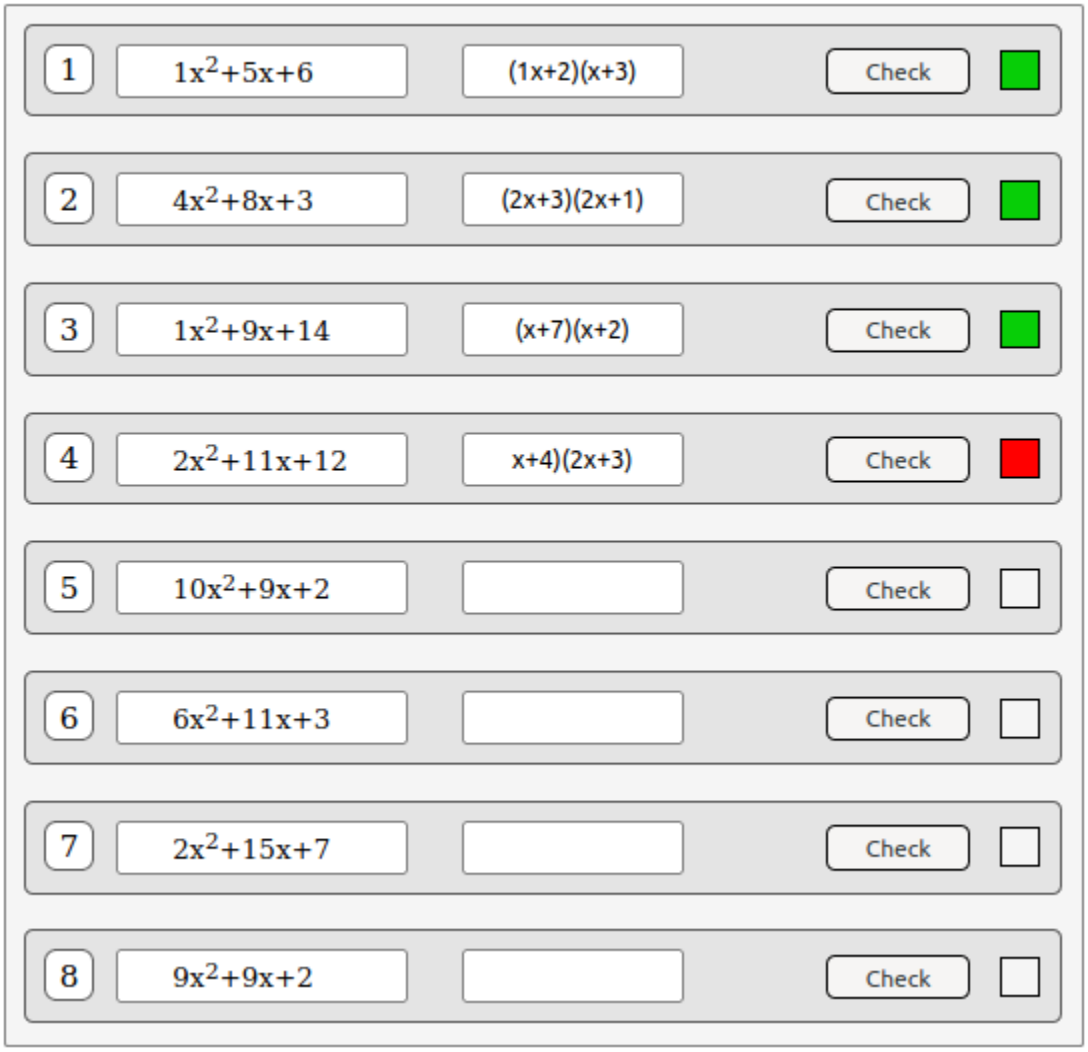
Factor By Grouping Problems



Summary

In this project a web application was created using HTML, CSS, and JavaScript. The intention of the project was to mimic a Factor by Grouping problem set that you would see in a standard math textbook. The application was required to have a series of problems, a place for the user to enter an answer, a button to check the answer, and a box to indicate to the user if their answer was correct. Another important focus of this project was to make it robust enough so that it can eventually be integrated into a larger math tutorial website. This project showed me the importance of sketching the design, being deliberate about the structure of the page and adding flexibility to handle future features.

Key Features of the Application

Figure 1: Description of Problem Box Elements

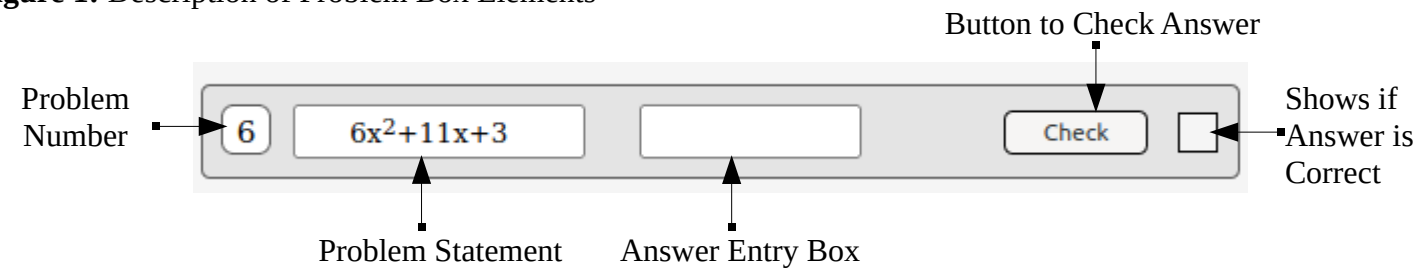


Figure 2: Some Permutations of Acceptable Answers

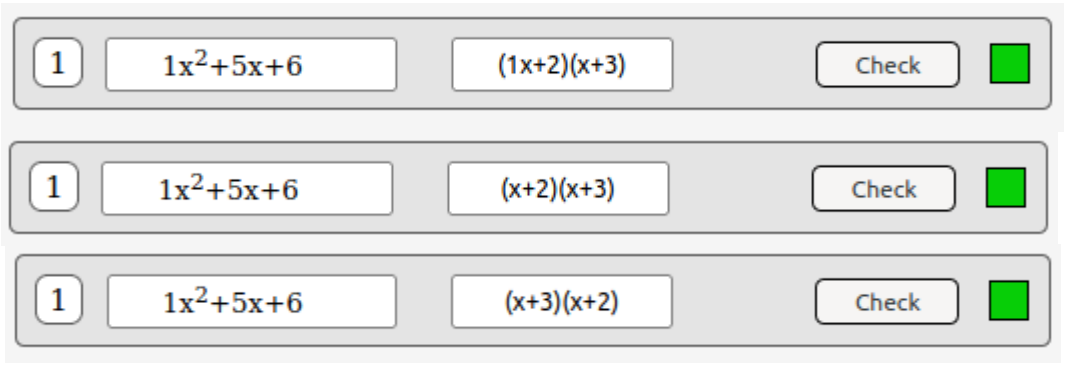


Figure 3: An example of an Improper Parenthesis

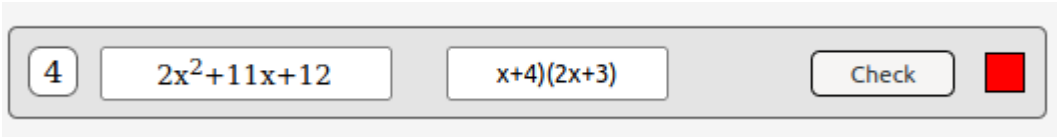
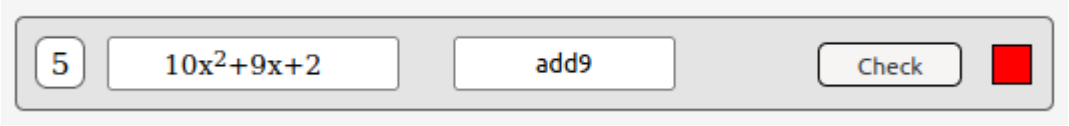


Figure 4: Handling of invalid Input



Description of Key Algorithms

The Generation of the Problem Set:
The important thing to realize about the Factor by Grouping method is that is it made up of distinct steps that are always the same. This is important for the generation of the problem set because when you consider each of those those steps, you realize that tackling the problem from one direction is easier than many other directions.

For example: going from the expression: x^2+5x+6 to its factor form: $(x+2)(x+3)$ is difficult. One of the intermediate steps in this process is finding factors of the product of the first and last terms. For small numbers this is not much of an issue, however some number may have a lot of factors and it would be prohibitively complex to try them all. That led me to seek out a better approach.

Well, going from the factored form: $(x+2)(x+3)$ to the original form: x^2+5x+6 is relatively easy. The process can be reduced to five steps that will be the same irregardless of what the coefficients are in the factored form. Therefore we can generate the HTML representation of each of the steps and save them to a list to access them later if we need to. For example, the problem we want to display to the user is the last step in that process which is just corresponds to the last element of the list. This makes the generation of the problem set much easier.

One last consideration may be: "Why don't you just generate random coefficients a, b, c and place them in the quadratic form: ax^2+bx+c*?* ". This may seem like the easiest method, however there is no guarantee that random expression that fits the quadratic form will actually be factorable. In this context, in order to be factored we need to have integer coefficients. Therefore answers such as $(\sqrt{2}x+3)(x+1)$ or $(x+1)(i+2)$ would be considered invalid. Now, it is true that we could use the quadratic formula to generate those answers but that is not in line with the theme of our original issue. It is assumed that an instructor would want to be deliberate and have control of the problems assigned to students. Therefore random coefficients, even if they produce a factorable result, will not suffice.

Determining if the User Input is a Correct Answer:
Checking the answer is more complex than it seems. This is due to the fact there are multiple correct answers to a Factor by Grouping problem. Consider the expression: x^2+5x+6 the factored form $(x+3)(x+2)$ is correct. However, due to the communicative property of multiplication: $(x+2)(x+3)$ is also correct. There is even more to this though, since convention dictates that there is an implied 1 in front of an *x* by itself. Therefore in order to check correctness, we would have to check every possible permutation of the answer. I felt that this would be overly complicated and that there had to be a better way.

An important moment was when I realized that we already had a function that could help us out with this. The code already had function that took in a list of coefficients from the factored form and returned a list of the steps to the factor by grouping method. Therefore instead of checking every permu-

tation, we could just extract the coefficients from the users answer and pass them through the function mentioned above. If the last step of the user solution matches the last step of the problems solution, then we say they are the same and that the user is correct. As long as the coefficients are exacted correctly, this will be true, regardless of the ordering.

Note: The above statement is only partially true. One could argue that: $(3+x)(2+x)$ is also a correct answer. This may be true, but it doesn't quite fit the *standard* math curriculum. For example: when foiled, $(3+x)(2+x)$ will yield $3x+6+x^2+2x$. If you use the associative law of addition, combine and rearrange terms it would still be *equivalent* to the *correct* answer x^2+5x+6 . This is interesting not because of the limited correctness of the algorithm but because of the reliance on convention. You need to draw the line somewhere, but its something to keep in mind for future versions.

There is one last thing to consider. What if the user entered an invalid answer? What if the user entered: (1 3)(x+2) or x+2)(x+3). This is problematic for the above logic because it assumes that the input is of the proper parenthesized form. Therefore before we check the answer we need to check if it is contains a valid parenthesis set. In this specific case, doing so is not that difficult since there is only one valid parenthesis "()()". Therefore we can iterate over the users answers, build a string, and if that string matches the parenthesis string, we have a valid case.

Web Application Code & Other Links

Html:
https://github.com/ryanA998/Factor-By-Grouping-Problems/blob/main/Factor_by_Grouping_Problem_Set.html

JavaScript:
https://github.com/ryanA998/Factor-By-Grouping-Problems/blob/main/Factor_by_Grouping_Problem_Set_Script.js