Create a new console app(.NetFramework) project for this project.

Fun with Methods

1. Let's create class to house our methods that we will be creating in this section and call it "CalculatorMadness". No need for any member variables or constructor.
2. Let's next create a method inside our new class like this:

```
0 references
public void AddTwoNumbers()
{
    Console.WriteLine("Please enter the first number:");
    int numberOne = int.Parse(Console.ReadLine());

    Console.WriteLine("Please enter the second number:");
    int numberTwo = int.Parse(Console.ReadLine());

    int result = numberOne + numberTwo;
    Console.WriteLine("The result of this addition is: "+ result);
}
```

3. This method as it is currently is relatively dynamic. Lets make this a bit more versatile though. Let's change this method to instead take in numberOne and numberTwo as parameters.

    a. When we change the method to take in the values instead. We no longer need to ask for user input in this method. Rather the variables will be assigned values from where the method is called.

4. Let's create another method in our class called "RunCalculations". Inside this method we will call our "AddTwoNumbers" method and pass it the values it is looking for.

    a.
    ```
    public void RunCalculations()
    {
        AddTwoNumbers(5,7);
    }
    ```
    b. When we move the variables to parameters, this allows the method to be used in more ways now. It isn't specifically based on user input anymore; it can now be used any time you want to add two values together.

5. We still have more work to do on this method. For example, this method currently just outputs the result to the console. Which means we can't do anything further with our result. Let's change our AddTwoNumbers to instead return our result back to where it was called.
    a. You will need to change the return type of the method to now reflect the new return type.
    b. Now that we have made this method take in two parameters and return the result. We have made this method much more flexible. It can be applied in a much larger spectrum

of situations. We can still call the method using user input if we want. But now we can use it for any and all two-number addition.

c.

```
public void RunCalculations()
{
    Console.WriteLine("Enter first number to add:");
    int value1 = int.Parse(Console.ReadLine());
    Console.WriteLine("Enter second number to add:");
    int value2 = int.Parse(Console.ReadLine());
    int output = AddTwoNumbers(value1, value2);
    Console.WriteLine("The result is: " +output);
}
```

   i. Example of passing user entered values into out add method^

6. Using only the Add method, we will call this same method multiple times to add more numbers together than just two.
   a. First, we will want to add 8 + 40. Then we will want to add the result of that to 200 + 50.
      i. (8+40) + (200+50)
   b. Capture the result and print it to the console.
      i. You cannot use the + operator in your RunCalculations method.
7. Next, we will create 4 new methods very similar to our Add method.
   a. Create a method to subtract one number from another number.
   b. Create a method to multiply two numbers.
   c. Create a method to divide one number by another number.
8. Using only the RunCalculations method and math methods you created. Have your logic solve the following math problem:
   a. $6+5-40*35/4+2^2$
   b. Your output should be –335
      i. Keep in mind order of operations!
      ii. $6 + 5 - ((40*35)/4) + 2^2$