Using Out

We can use `out` parameters in our own methods as well. In this example, `Yell()` converts `phrase` to uppercase and sets a boolean variable to `true`:

```
static string Yell(string phrase, out bool wasYellCalled)
{
  wasYellCalled = true;
  return phrase.ToUpper();
}
```

- The `out` parameter must have the `out` keyword and its expected type
- The `out` parameter must be set to a value before the method ends

When calling the method, don't forget to use the `out` keyword as well:

```
string message = "garrrr";
Yell(message, out bool flag);
// returns "GARRRR" and flag is true
```

☑Instructions

1.

Declare a method `Whisper()` with a `string` parameter and `out` `bool` parameter. It should return a `string`.

Hint

The declaration for `Whisper()` should look like:

```
static string Whisper(string phrase, out bool wasWhisperCalled)
```

2.
Define the method body. `Whisper()` should work like `Yell()`, but instead of returning an uppercase string, it returns a lowercase string.

Once defined, you should be able to call it like:

```
string statement = "GARRRR";
Whisper(statement, out bool marker);
// should return  "garrrr" and set marker to true;
```

Hint

You'll need to use the `ToLower()` method:

```
string statement = "GARRRR";
statement.ToLower(); // returns "garrrr"
```

And you'll need to set the `out` `bool` parameter to `true`.

3.
Call `Whisper()` in the `Main()` method and print the returned value to the console.

Make sure to use an `out` modifier when calling the method!