Basic Method Syntax: Select

To transform each element in a sequence — like writing them in uppercase — we can use the select operator. In method syntax it's written as the method Select(), which takes a lambda expression:

```
string[] heroes = { "D. Va", "Lucio", "Mercy", "Soldier 76", "Pharah",
   "Reinhardt" };
var loudHeroes = heroes.Select(h => h.ToUpper());
```

We can combine Select() with Where() in two ways:

Use separate statements:

```
var longHeroes = heroes Where(h => h.Length > 6);
var longLoudHeroes = longHeroes.Select(h => h.ToUpper());
```

Chain the expressions:

```
var longLoudHeroes = heroes
.Where(h => h.Length > 6)
.Select(h => h.ToUpper());
```

As with most of LINQ, the choice is up to you!

In the first option, we use two variable names and two statements. You can tell there are two separate statements by counting the semi-colons.

In the second option, we use one variable name and one statement.

If we must use method-syntax, we prefer the second option (chaining) because it is easier to read and write. You can imagine each line like a step in a conveyor belt, filtering and transforming the sequence as it goes.

✓Instructions

1.

Write a method-syntax query that:

Selects all of the elements in heroes containing the character "c" and Transforms them to lowercase

Use separate statements and store the final result in a variable lowerHeroesWithC.

Hint

Here's an example method-syntax query with separate statements:

var shortBirds = birds.Where(b => b.Length < 7);
var shortBirdsInCaps = shortBirds.Select(b => b.ToUpper());

When writing your own query:

Use Where() and Contains() to find each element containing "c".

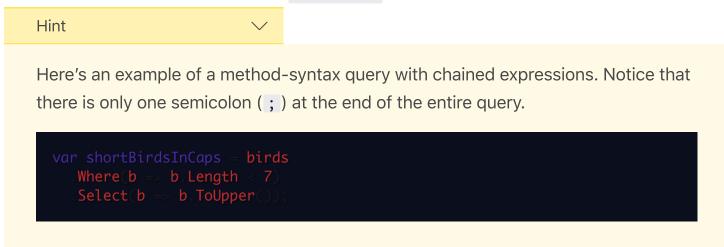
Use Select() and ToLower() to transform each string.

Use two variables: one for the first statement and one for the second.

2.

Write a second query that does the same thing, but uses chained expressions.

Store the result in a variable named sameResult.



3. Print out all of the elements of sameResult to check your work.

