Expression-bodied Definitions

Expression-bodied definitions are the first "shortcut" for writing methods. They're great for writing one-line methods, like this one:

```
bool IsEven(int num)
{
  return num % 2 == 0;
}
```

We can rewrite this definition as an expression-bodied definition by:

- removing the curly braces and `return` keyword, and
- adding the "fat arrow", or `=>`, which is composed of the equal sign, `=`, and greater than, `>`, symbols

```
bool isEven(int num) => num % 2 == 0;
```

This also works for methods that return nothing, aka `void`:

```
void Shout(string x) => Console.WriteLine(x.ToUpper());
```

This type of definition can only be used when a method contains one expression. This helps us remember the name: *expression*-bodied definitions are method definitions with one *expression*.

Fun fact: some developers also call the fat arrow notation, `=>`, a squid! 🦑

☑Instructions

1.
Convert the method `DaysToRotations()` to an expression-bodied definition.

Hint

As an example, this definition:

```
bool IsEven(int num)
{
  return num % 2 == 0;
}
```

can be written as:

```
bool isEven(int num) => num % 2 == 0;
```

2.
Convert the method `Welcome()` to an expression-bodied definition.

Hint

As an example, this definition:

```
void Shout(string x)
{
  Console.WriteLine(x.ToUpper());
}
```

can be written as:

```
void Shout(string x) => Console.WriteLine(x.ToUpper());
```