While Loop

Loops are used to repeat a set of instructions based on a set of conditions. If this makes you think of conditional statements, you're not wrong!

The `while` loop looks very similar to an `if` statement. Just like an `if` statement, it executes the code inside of it if the condition, a statement that evaluates to a boolean value, is true.

```
while (condition)
{
  statement;
}
```

However, unlike an `if` statement that runs once, a `while` loop will continue to execute the code inside of it, over and over again, *while* the condition is true. The computer is constantly checking to see if the condition is satisfied. For this reason, `while` loops are useful when you know at what point a program should stop, rather than the number of times it should repeat.

In your video game, you want the player to rise up in the air as long as the user is pressing the spacebar:

```
while (spacebar == "down")
{
 RiseUp();
}
```

In this example, while `spacebar` is equal to `down`, the character will keep rising on the screen. It will exit the `while` loop once the user releases the spacebar and then the rest of the program will continue.

`while` loops can get dangerous, because they depend on that condition to at some point return false. What if we never take our finger off the spacebar? Sounds ridiculous, but theoretically, the program would run forever! This is known as an *infinite loop*. If you get stuck in an infinite loop while on Codecademy, you can reload the page to stop it.

☑Instructions

1.

You're really trying to step up your "inbox zero" game and want to build a tool that can help you get there. For your first prototype, you're just going to delete all the emails.

Write a `while` loop that checks to see if there are any emails in your inbox. If there are still emails, decrease the amount of emails by one until there are no emails left.

Hint

A `while` loop will continue running until a stopping condition is met:

```
// start with zero strawberries
int strawberries = 0;

// this will stop looping once
// 10 strawberries exist
while (strawberries < 10)
{
  strawberries++;
}
```

2.

It's a little hard to tell what's happening in our program, so within the loop have it print a message that it's deleting an email and how many emails are left.

Hint

Adding a `Console.WriteLine()` statement inside a loop with a variable can help us keep track of changing values:

```
// start with zero strawberries
int strawberries = 0;

// as long as there are less than 10 strawberries,
// add one to the container
while (strawberries < 10)
{
  strawberries++;

  // Print a message each time a strawberry is added
  Console.WriteLine($"adding a strawberry. Total strawberries =
{strawberries}");
}
```

3.

When your inbox reaches zero, have your program print out “INBOX ZERO ACHIEVED!” or some other congratulatory message.

Hint

After a `while` loop completes, the rest of the program will continue:

```
// start with zero strawberries
int strawberries = 0;

// as long as there are less than 10 strawberries,
// add one to the container
while (strawberries < 10)
{
  strawberries++;
  Console.WriteLine($"adding a strawberry. Total strawberries =
{strawberries}");
}

// after 10 strawberries are added, print a message:
Console.WriteLine("Container full!");
```