Homework 2 - Easter Egg Hunt

Due: Monday February 22nd at 5:00 PM

You have decided that easter isn't just for little kids. Great programmers such as yourselves also hunt eggs. However you DREAD the idea of getting up from your chair, listening to your little cousins, and worst of all: going outside. Thus, you decide you are just going to write a VB program to simulate Easter's best part, the chocolate.

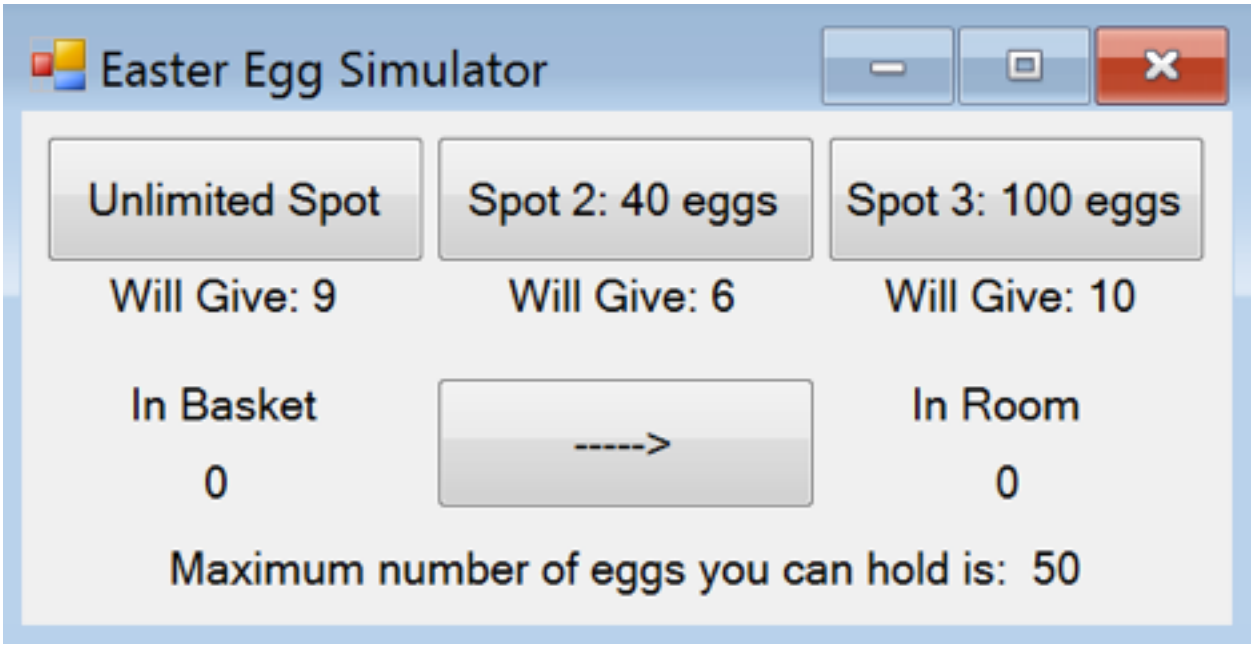
Your program is going to simulate the process of going to different places around the property (upstairs, downstairs, front lawn, playground, etc, name them what you want). Each of the **THREE** spots will be a separate button on a form and you "visit" it by simply clicking on the button. When collecting eggs you put them in your basket; however, your basket can only hold **50** eggs at any one time. To collect more eggs you must drop them off in your room (one more button). This button transfers all the eggs in your basket to your room, which can hold infinite eggs.

1. Spot one, the easiest to find (and complete) has an **unlimited** supply of chocolate eggs. Every time you click you should get **9** eggs.
2. Spot two only has **40** eggs in it. Each time you visit you should take **6** eggs. Don't go negative, negative eggs don't exist! Make sure to take the remaining eggs when there are less than 6 left (assuming you have room).
3. Spot three has **100** eggs available but you shouldn't be greedy here. Take **10** eggs the first time, then **9** eggs, then **8** eggs, and so on. Make sure that you don't start adding eggs to the pile.

Whenever you are about to receive eggs make sure you have the room to accept it and the spot still has eggs to give!

This homework reviews the concepts of variables, constants, scope, conditional statements, and use of variables to keep track of state. Your held eggs and your stored eggs should be implemented with global/module level variables. Each of the houses' candy pools can EITHER be implemented as **static** variables or as **global/module** level variables. Do not allow the eggs to go negative or create eggs out of nothing. You should use **constants** for the amounts of eggs each spot will give away.

Here is a sample of what your form could look like:



(You can make it fancier if you want with images or Easter colors. Do **not** make it garish or hard to read with bad color choices or messy/distracting images)

Static Variable Example

Note: this is not required for the homework but is a **much** cleaner way to do it. As a general rule you should **avoid global variables <u>whenever</u> possible**.

If you want to use static variables for your homework to keep track of the house's candy pools this example will help you understand what a static variable is and how it differs from a normal variable.

Copy and paste the following into the button click sub procedure for a button named "btnExample".

```
Dim myNormalCounter As Integer = 100
myNormalCounter = myNormalCounter - 1
btnExample.Text = Convert.toString(myNormalCounter)
```

Try to click the button a few times. You will notice that the button will show 99 forever. Now try changing the "Dim" to either "Static Dim" or just "Static" and click the button.

```
Static myNormalCounter As Integer = 100
myNormalCounter = myNormalCounter - 1
btnExample.Text = Convert.toString(myNormalCounter)
```

Now it will count down from 99 one at a time. Why?

———

A normal variable has a lifetime (called it's scope). When the sub procedure that created the variable ends, the variable is destroyed. The next time the sub is called the variable is created again with its default value (in this case 100).

A static variable is just like a normal variable with one crucial change. It's lifetime is extended, it will never be reinitialized. The first time you call the sub the variable is created and initialized to the value specified. The next time you call the function, the variable still exists and will not be reset/reinitialized to the value specified.

It is important to see that static variables are similar to module level variables in that they are only created once (never reset). However they are different in that you can only access them inside of their scope, unlike module level variables which can be accessed anywhere. This is useful in that you can use the same name in different places to refer to different variables, just like local variables. Additionally you wont pollute the global space. If any of this confuses you, come to office hours!