Core C# Casting, Operators, and Flow Control Lab

Now that we know some language basics for C#, let's put them to practice. We'll work with a couple of business objects and add some real-world logic to them.

Creating a Cart Subtotal method

A shopping cart contains multiple line items – different quantities of different products. We should create a cart class and be able to get a subtotal from it.

1. Create a Cart class if you don't have one already. If you do, open it.
2. It should have a property like this:

```
public List<OrderDetail> OrderDetails { get; set; }
```

3. Write a new method called Subtotal(). It should return a decimal representing the subtotal for the cart.
4. The subtotal should foreach through the OrderDetails in the cart, multiply each quantity by its Product.UnitPrice and sum the products together. (Hint: You may need to do some converting of data types to do the math).
5. Build it to make sure it compiles. Don't worry about testing it yet. We can do that later.

Validating Credit Cards

Now, let's make sure that any entered credit cards are valid. There is a pattern that all credit cards must match. It is called the Luhn Algorithm. It works like this:

- Create a "total"
- Add all the even digits (2nd, 4th, 6th, 8th, etc) to total.
- For each odd digit (1st, 3rd, 5th, 7th, etc), add to total according to this:
  - If the digit is 0, add 0.
  - If the digit is 1, add 2.
  - If the digit is 2, add 4.
  - If the digit is 3, add 6.
  - If the digit is 4, add 8.
  - If the digit is 5, add 1.
  - If the digit is 6, add 3.
  - If the digit is 7, add 5.
  - If the digit is 8, add 7.
  - If the digit is 9, add 9.
- If total is divisible by 10, it is valid.

6. Create or edit your CreditCard class. Implement a method called CreditCard.IsValid() which should return a boolean; true if the card passes and false otherwise. (HInt: You can get to digit X by creditCardNumber[X])

Testing it

Let's do a little old-school testing of our Luhn Algorithm. This is not the best way to test it, but we'll learn the proper way later.

7. In Solution Explorer, right-click on the Windows project and choose "Add Reference ...". Then choose the Core project under the "Projects" tab.
8. Open the default Windows form that was automatically added. Open the Toolbox (hint: it is usually found on the left side of Visual Studio).
9. Drag a textbox and a button onto the form.
10. Double-click the button to create a Click event handler.
11. At the top of the code, add this:

```
using Core;
```

12. Then in the button click event, add this:

```
var card = new CreditCard()
{
  Number = textbox1.Text
};
```

13. Then call card.IsValid(). Tell the user the result like this:

```
MessageBox.Show(string.Format("Card is {0}", result);
```

14. Test your credit card validation by entering in a couple of card numbers that you know to be valid and invalid.
15. Email all the valid ones to your instructor. (Just kidding).