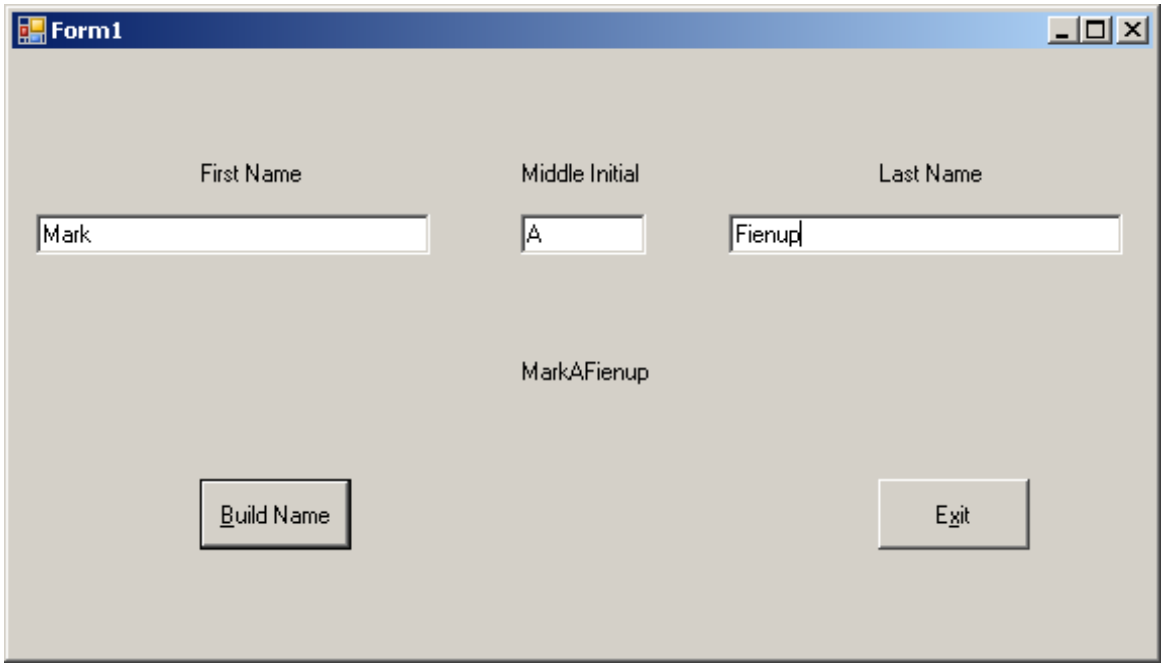
Name:_____

1) The code btnBuildName.Click is not quite right since it does not put spaces between the parts of the name. How would you fix the code?

```
Private Sub btnBuildName_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles btnBuildName.Click

    lblFullName.Text = txtFirstName.Text  &  txtMiddleInitial.Text  &  txtLastName.Text

    lblFullName.Visible = True
End Sub
```



2) A *variable* is a named spot in memory that the programmer can use to store a value. In VB, variables must be *declared* explicitly by the programmer and causes the variable to be created. The format/syntax of a variable declaration (called a *Dimension statement*) is:

Dim variableName As DataType

where Dim and As are keywords, the variableName is a meaningful name chosen by the programmer (rules: 1st char. is letter or '_', and the rest can be letters, '_', or digits), and DataType is a VB data type:

- an integer type: Byte, Short, Integer, Long
- a floating-point (real type): Single, Double, Decimal
- some other common data types: Boolean, Char, String, Date

In the above btnBuildName.Click code, suppose we wanted to declare a string variable in which to store the full name before we displayed it.

a) What would be meaningful variable name?

b) What would the Dimension statement look like for this variable?

c) Write the assignment statements to build the string and then assign it to lblFullName.Text.

3) Given the operator precedence for VB's mathematical, Boolean, and logical operations is (from highest to lowest):

- Operations that are enclosed in parentheses.
- Exponentiation (^)
- Unary negation (-)
- Multiplication (*), floating point division (/)
- Integer division (\)
- Modulus remainder (Mod)
- Addition (+) and subtraction (-)
- String concatenation (&)
- Relational operators (=, <>,<, >, <=, >=)
- Not
- And
- Or
- Xor

Operators within each level are performed left-to-right. Evaluate each of the following:

a) 6 + 3 * 5

b) (6 + 2) \ 3

c) 4 + 2 ^ 3 - 5

d) 7 Mod 4 + 5 * 6

e) (6 + 2) / 3

4) *Implicit type conversion* occurs when you assign a value of one type to a variable of a different data type. In chapter 1 (Figure 1-15) when we were setting up the Visual Basic programming environment, we set Option Strict to "On". This has the effect of only allowing *widening* conversions so that numeric accuracy is less likely to be lost. For integer numeric types, the following implicit numeric conversions are allowed with Option Strict "On":

Byte 8-bit unsigned	Integer 32-bit signed	Long 64-bit signed -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	Decimal (29 significant digits) Used in financial calculations	Single (7 sign. digits) ~1.0×10^{38}	Double (15 sign. digits) ~1.0×10^{308}

Allowed widening

Even with Option Strict "On", give some examples of when numeric accuracy is lost.

Name:_____

5) When we convert a numeric value to a string, you can use a formatting string in the ToString function. Complete the following table:

Number Value	Format String	ToString( ) Value
12.3	n4 (number 4 decimal places and commas)	
12.348	n1 (number 1 decimal place and commas)	
1234567.1	n (number default of 2 decimal places and commas)	
1234.560	f1 (fixed-point 1 decimal place)	
123456.0	e3 (exponential: normalized so 1 digit is to left of decimal point and 3 to right)	
.234678	p2 percent with 2 decimal places and '%'	
-1234567.8	c3 (currency with $ and 3 decimal places)	

Homework #1
Due: Sept 12, 2009 (Saturday at 11:59 PM)
Chapter 3 Programming Challenge 3. Test Score Average