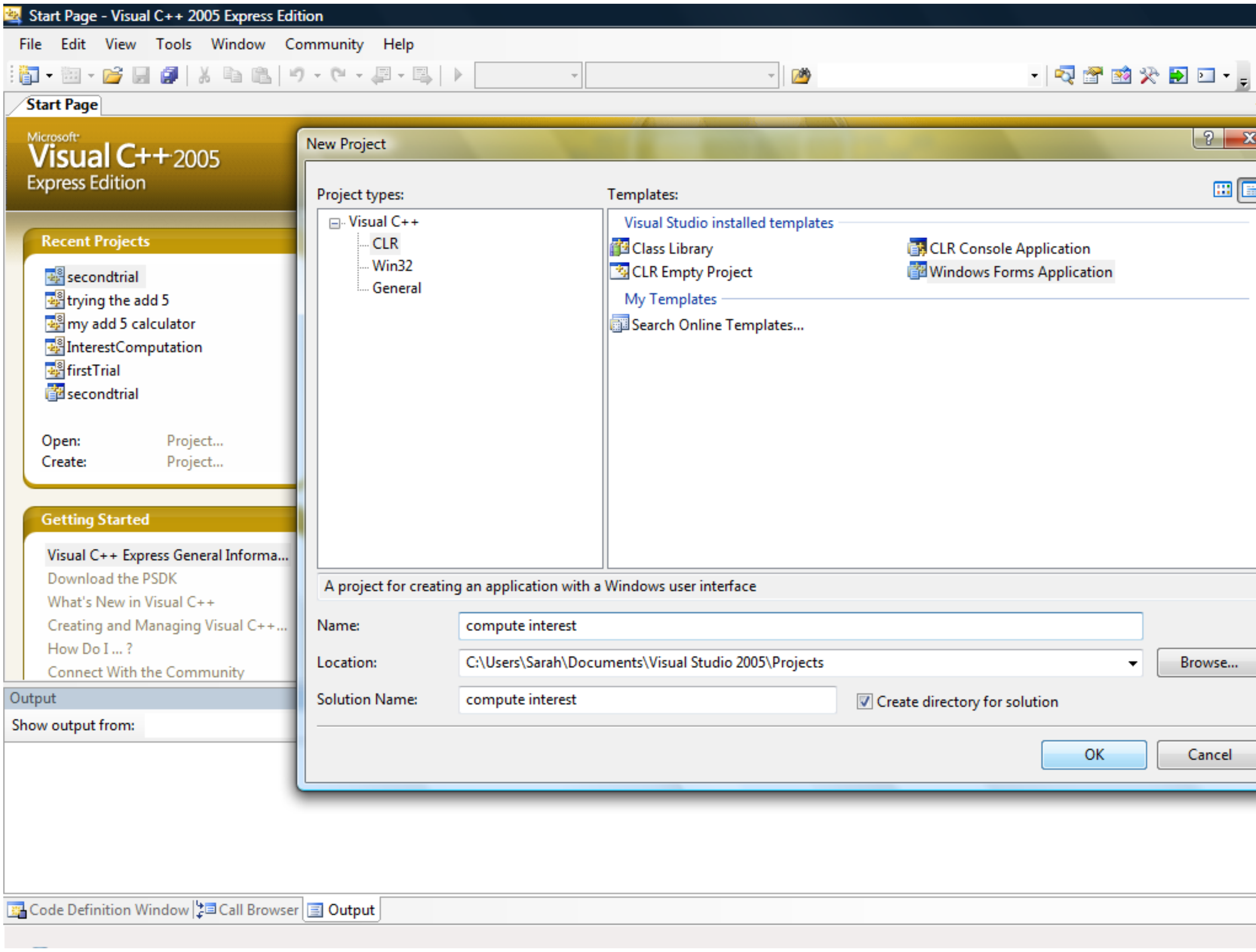
Computing Interest Using MFC

Much of the user friendly software we see today has a graphical user interface (GUI, pronounced gooey). This is the windows desktop that we have become familiar with. If we supply a GUI, users can click on buttons and enter text into boxes for input. The steps below will help you to complete the Interest Computation Program using GUIs.

a) **Open a new project choosing CLR (Microsoft's Common Language Runtime protocols) as the Project type. Choose Windows Forms Application under Templates. Below it is called compute interest.**



A project for creating an application with a Windows user interface

Name: compute interest

Location: C:\Users\Sarah\Documents\Visual Studio 2005\Projects

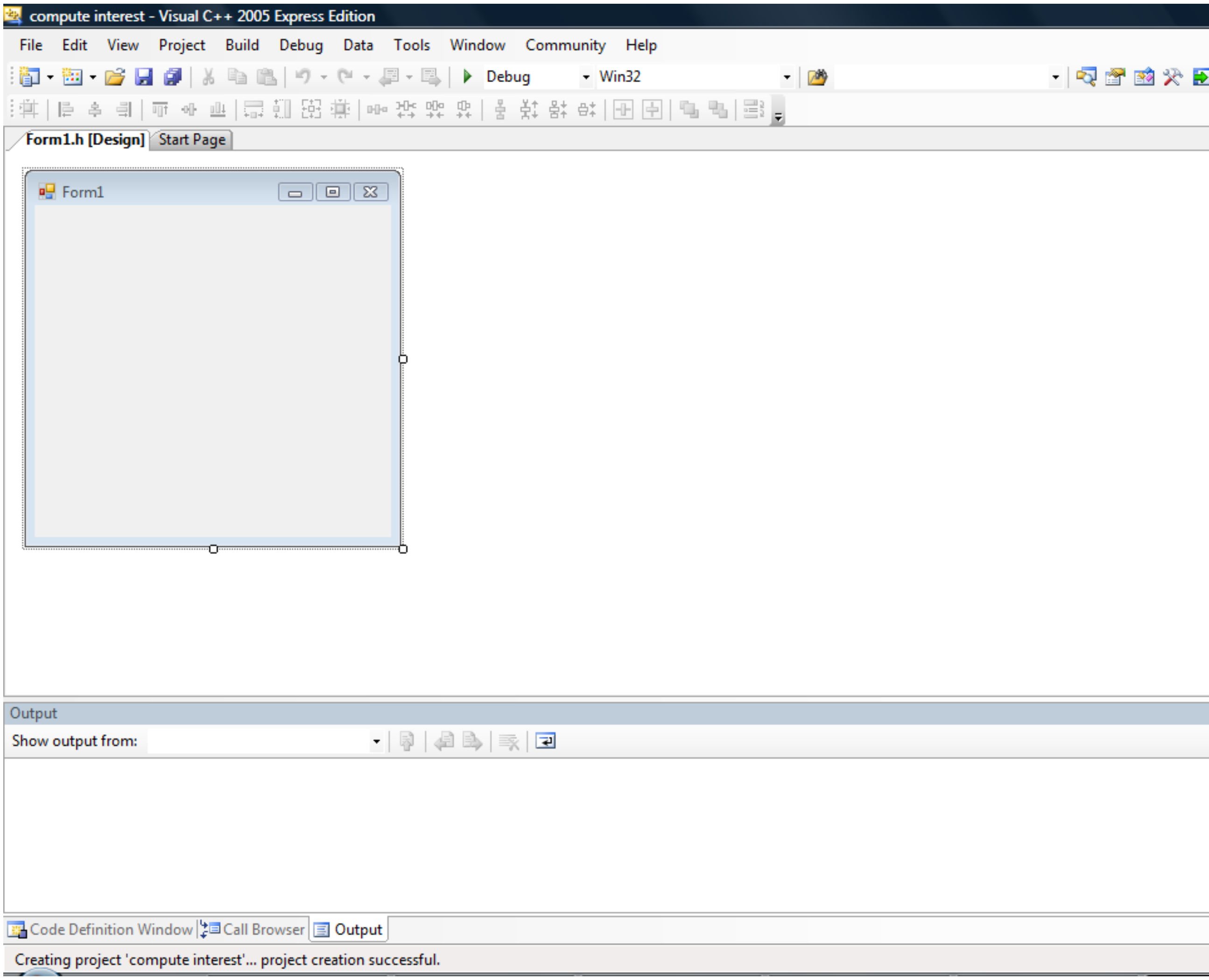
Solution Name: compute interest

☑ Create directory for solution

b) If you wish your interface to be set up differently, you may use the view menu to open the solution explorer, properties, and the toolbox.

You can then dock all three windows. (See My first GUI example to get help with this).

c) Your screen should then look like this:

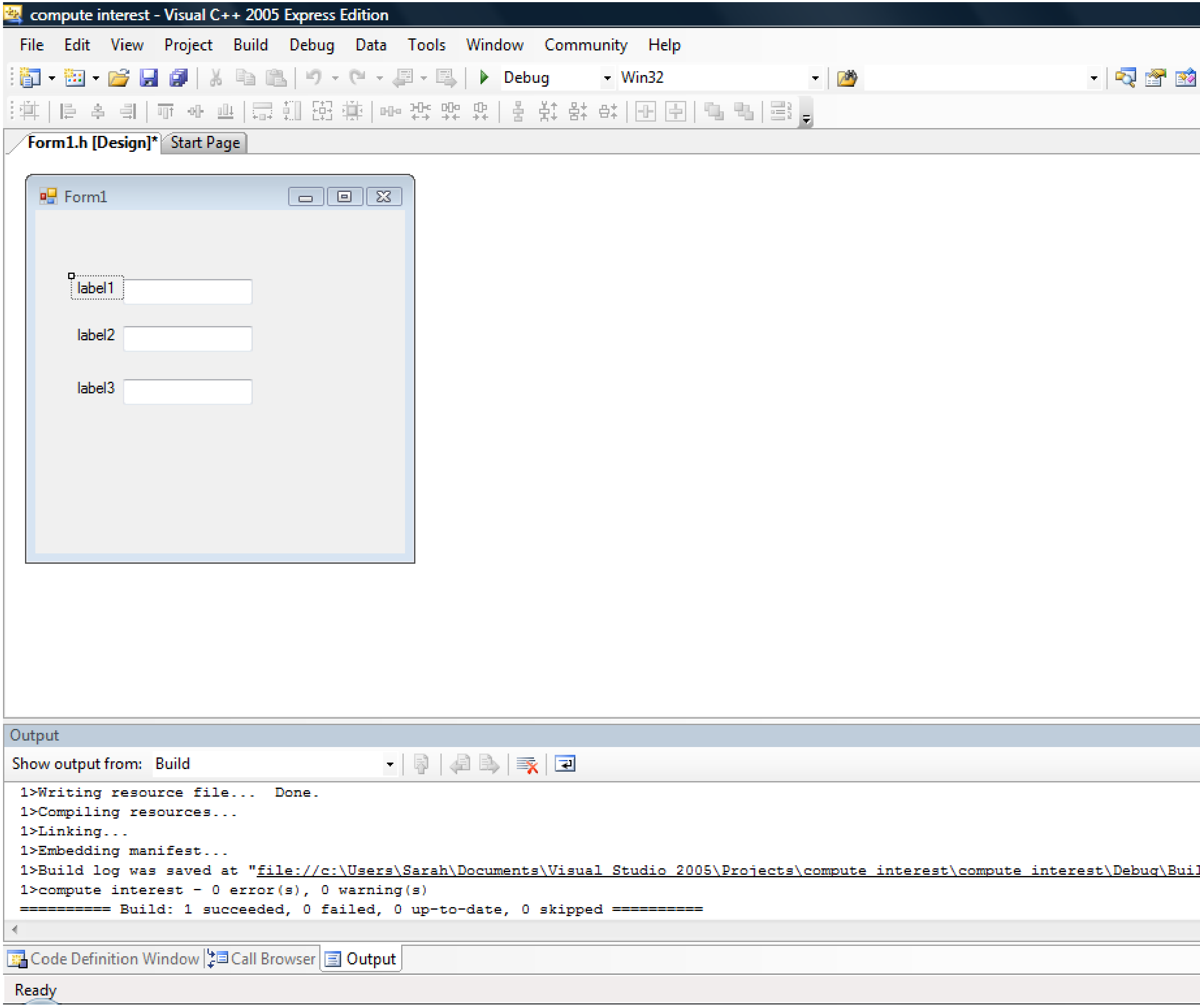


compute interest - Visual C++ 2005 Express Edition
File Edit View Project Build Debug Data Tools Window Community Help
Debug
Win32
Form1.h [Design]
Start Page
Form1
Output
Show output from:
Code Definition Window
Call Browser
Output

Creating project 'compute interest'... project creation successful.

d) You may link and run your program. When you run your program, an empty form comes up. Click on the close button at the corner to close that window.

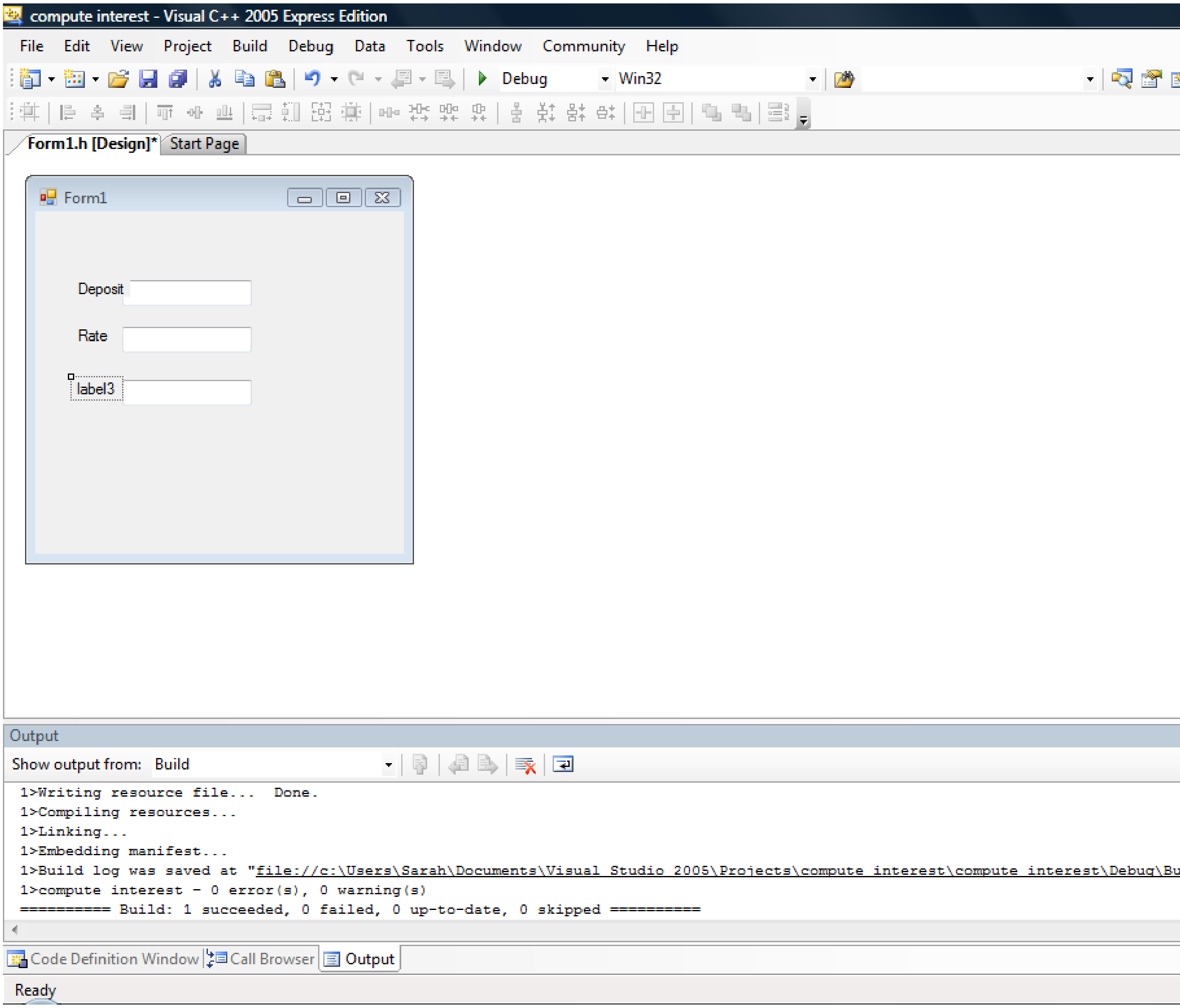
e) You can see the Toolbox at the right hand side of the screen above. Looking at *part a)* of the problem as described for the console application in *Part 1)* above, we wish to create input boxes for our three known variables in the interest computation program. This will allow the user to enter values. Drag three labels (**A**) and three text boxes (**ab|**) onto the form.



Show output from: Build

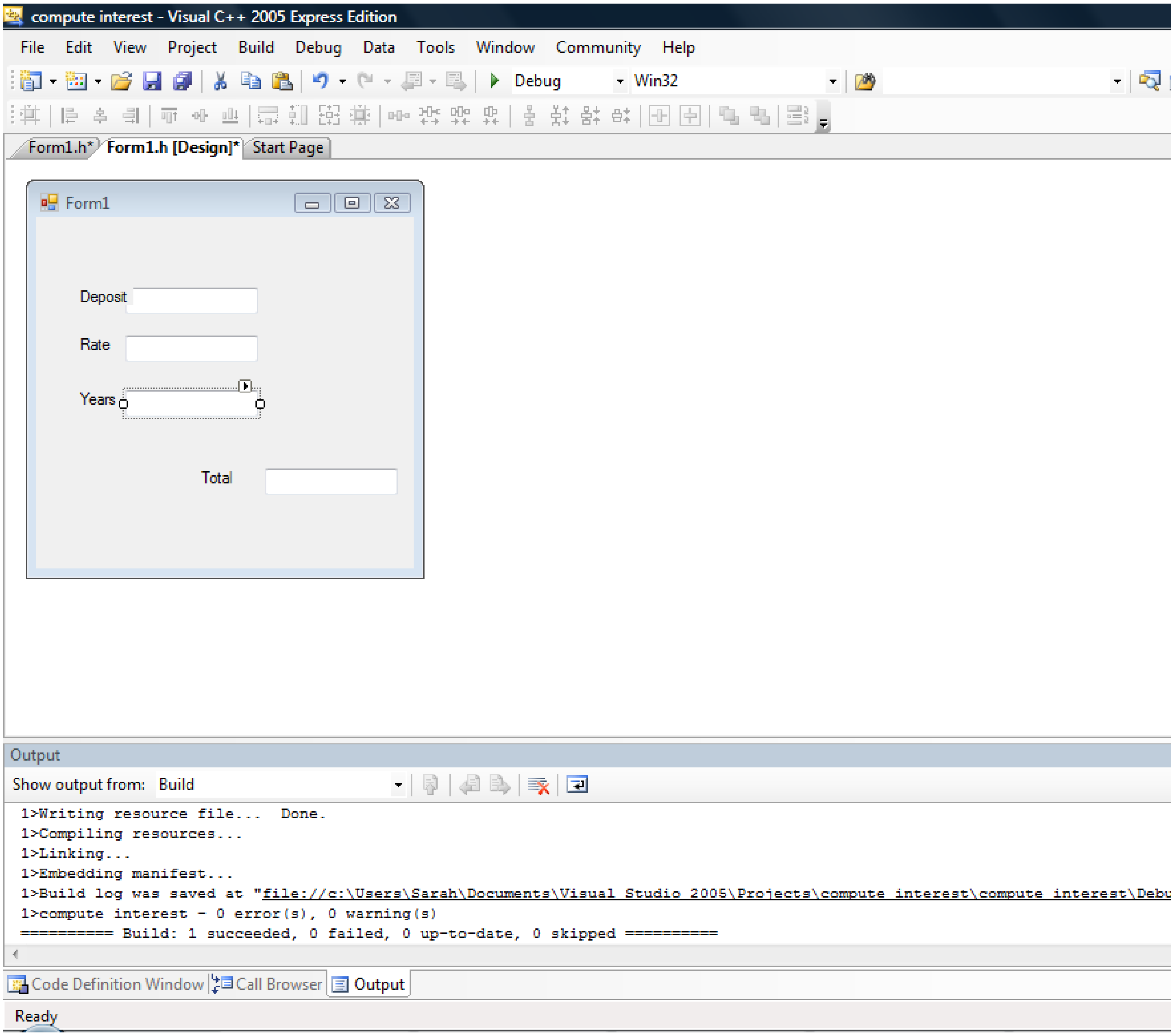
```
1>Writing resource file...  Done.
1>Compiling resources...
1>Linking...
1>Embedding manifest...
1>Build log was saved at "file://c:\Users\Sarah\Documents\Visual Studio 2005\Projects\compute interest\compute interest\Debug\Bui
1>compute interest - 0 error(s), 0 warning(s)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

f) For each of the label boxes: Select the label, then change the text of the label using the Properties window.



```
1>Writing resource file...  Done.
1>Compiling resources...
1>Linking...
1>Embedding manifest...
1>Build log was saved at "file://c:\Users\Sarah\Documents\Visual Studio 2005\Projects\compute interest\compute interest\Debug\Bu
1>compute interest - 0 error(s), 0 warning(s)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

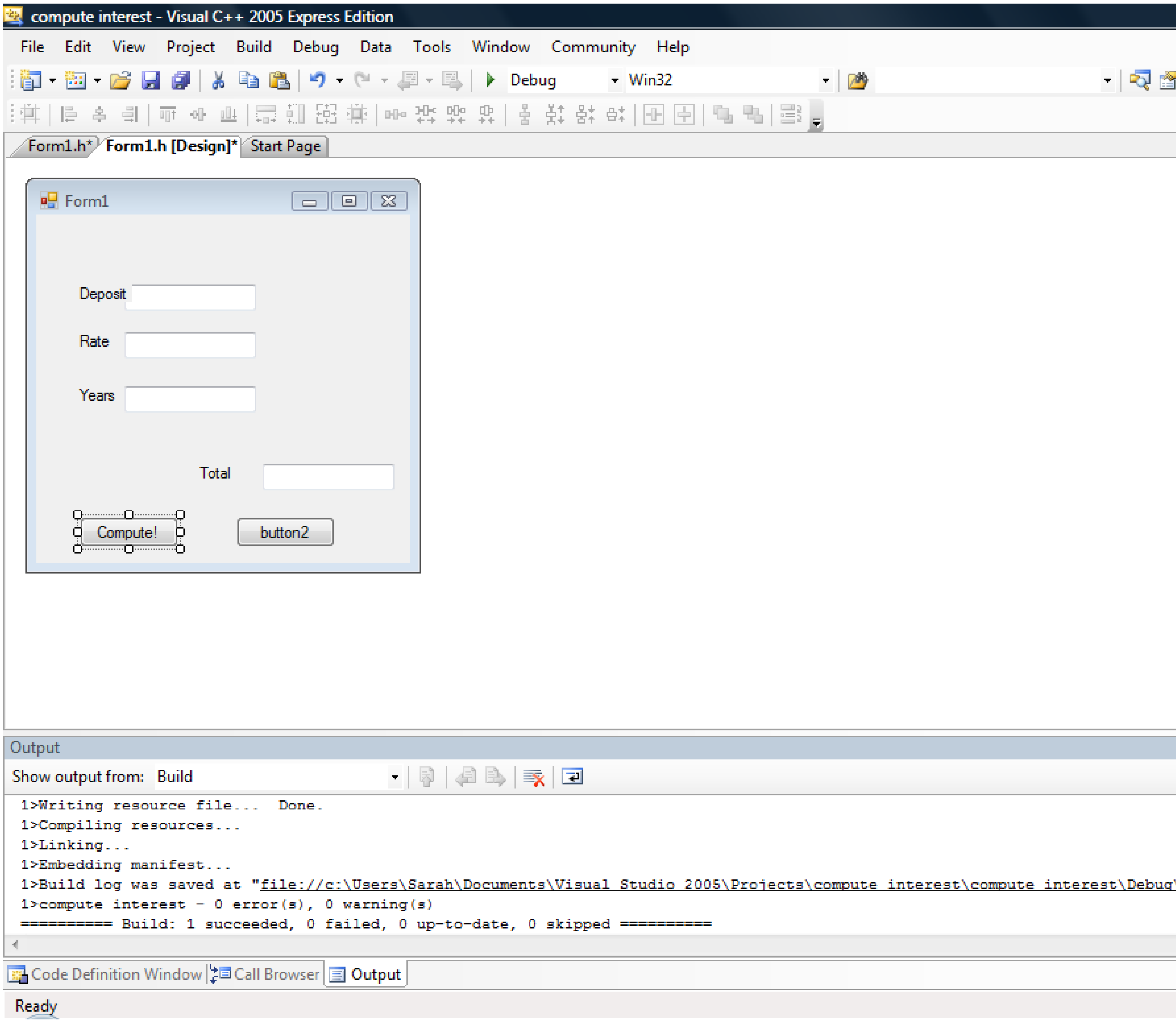
a) Place another text box, and label box for the output variable, changing the caption of the label box as above.
   Click on each of the text boxes, and change the name in the properties window. This will allow you to use a meaningful name in your program.
   Right click on each button, changing one caption to compute and one to clear. For these buttons, you should also change the ID to something more meaningful.
   The compute button will be used to calculate the final result, and the clear button will be used to clear all input values.



Show output from: Build

```
1>Writing resource file...  Done.
1>Compiling resources...
1>Linking...
1>Embedding manifest...
1>Build log was saved at "file://c:\Users\Sarah\Documents\Visual Studio 2005\Projects\compute interest\compute interest\Debu
1>compute interest - 0 error(s), 0 warning(s)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

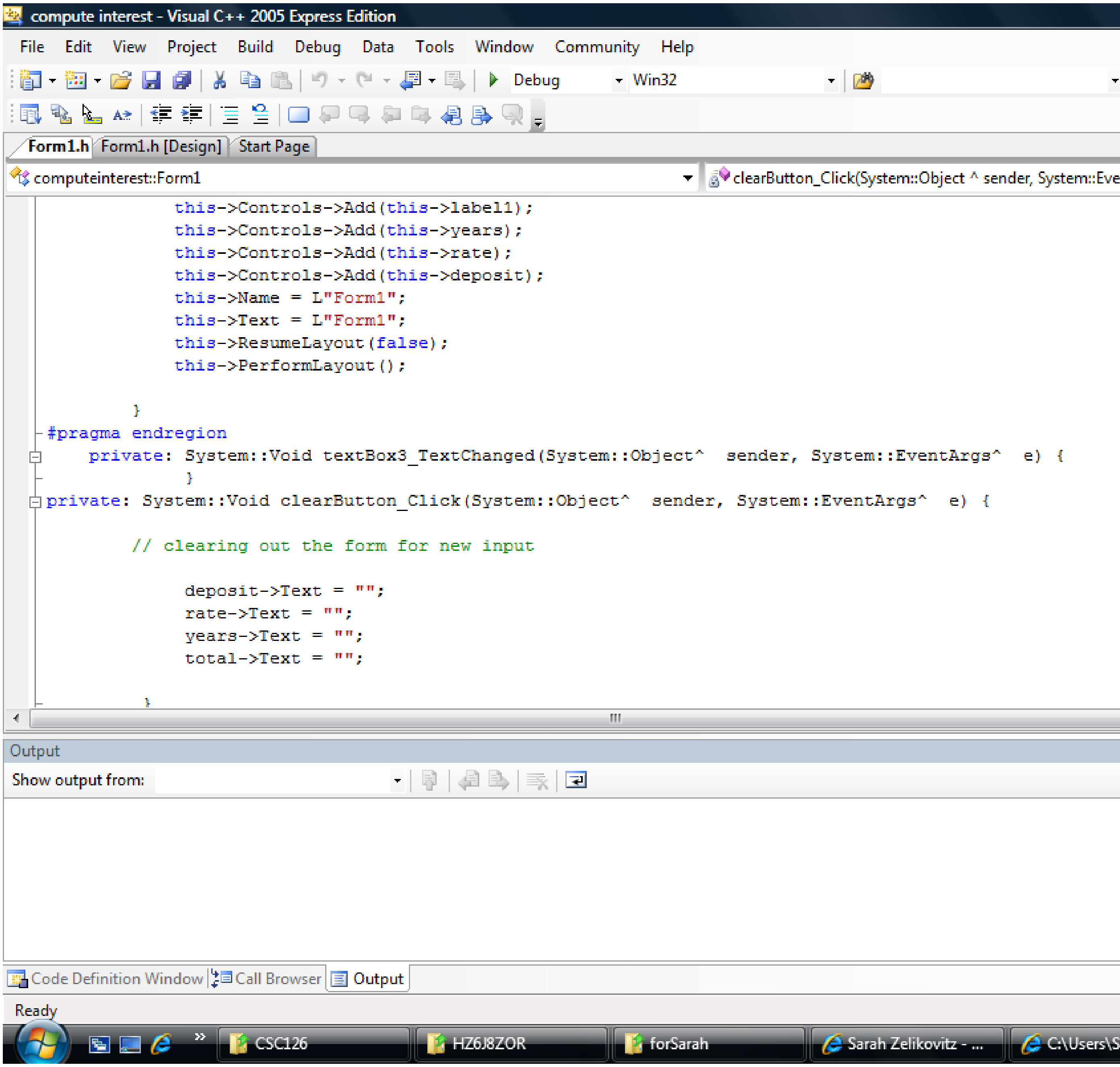
b) Drag two buttons from the Toolbox onto the form. Change the Text properties and the name of each of these buttons. You may “Run without Debugging” again to see your program Compile, Link and Execute.



Show output from: Build

```
1>Writing resource file...   Done.
1>Compiling resources...
1>Linking...
1>Embedding manifest...
1>Build log was saved at "file://c:\Users\Sarah\Documents\Visual Studio 2005\Projects\compute interest\compute interest\Debug
1>compute interest - 0 error(s), 0 warning(s)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

c) Double click on the Clear button to go to the code of the event handler to enter the C++ commands below.



```
				this->Controls->Add(this->label1);
				this->Controls->Add(this->years);
				this->Controls->Add(this->rate);
				this->Controls->Add(this->deposit);
				this->Name = L"Form1";
				this->Text = L"Form1";
				this->ResumeLayout(false);
				this->PerformLayout();

			}
#pragma endregion
	private: System::Void textBox3_TextChanged(System::Object^  sender, System::EventArgs^  e) {
				 }
private: System::Void clearButton_Click(System::Object^  sender, System::EventArgs^  e) {

			// clearing out the form for new input

				 deposit->Text = "";
				 rate->Text = "";
				 years->Text = "";
				 total->Text = "";

		 }
```

d) Double click the compute button to create the event handler for compute. You wish the calculation of interest to be performed each time the user clicks on the compute button. Modify your code from *Part I* of this assignment to use the user's input to set the value of the edit box that has not been filled in. Hint: First write the code that allows the user to place the starting amount, interest rate and years and have your program fill in the final balance. Then change your code so that the user can input ANY of the three variables, and have the fourth computed. Remember, to place a value from a text box into a double variable, you must use the TryParse function. For example:

```
     double initialDeposit;

     Double::TryParse(deposit->Text,initialDeposit);
//this function takes the text from the deposit textbox,
                                                             /
                                                        /con
                                                        vert
                                                        s it
                                                        to a
                                                        doub
                                                        le,
                                                        and
                                                        plac
                                                        es
                                                             /
                                                             /
                                                             i
                                                             t

                                                             i
                                                             n
                                                             t
                                                             o

                                                             t
                                                             h
                                                             e

                                                             v
                                                             a
                                                             r
                                                             i
                                                             a
                                                             b
                                                             l
                                                             e

                                                             i
                                                             n
                                                             i
                                                             t
                                                             i
                                                             a
```

lDeposit (see the add5 example for //more help on this)

How can you check to see which of the boxes is empty and must be computed?
You can compare a textbox to the empty string to see if it is empty:

```
if (deposit->Text == "")
```

but to avoid errors instead use:

```
if (deposit->Text->Trim() == "")
```

The expression above uses the `Trim()` function to remove all leading white spaces and ending white spaces from the text that is in deposit, before it compares the text to the empty string. This way, a use cannot enter spaces (looks empty but is not!) and cause the program to fail.

NOTE: TO CALCULATE THE ABOVE YOU SHOULD HAVE *e* DECLARED AS A `const double`, AND YOU MUST `#include <cmath>`. BOTH THESE LINES CAN BE PLACED AT THE BEGINNING OF THE MAIN HEADER FILE OF YOUR APPLIATION (LOOK UNDERTHE FILES TAB OF THE WORKSPACE WINDOW AND CHOOSE THE HEADER FILE WITH THE NAME OF YOUR APPLICATION).

e) Allow the user to choose one of two options: compounded continuously (as above) or compounded monthly. (total =). You can do this with a few more buttons as above, or explore the help files to use radio buttons.