Computer Science 127 - Lab Exercise 1

Introduction to Excel User-Defined Functions (pdf)

During this lab you will experiment with creating Excel user-defined functions (UDFs).

Background

We use a spreadsheet to perform repetitive calculations using formulas that we develop. Sometimes the formulas are straightforward arithmetic expressions and sometimes they are complicated formulas involving several arithmetic expressions as well as built-in functions such as AVERAGE, IF or VLOOKUP. Consider the profit and tax calculation spreadsheet discussed in class. Depending on how complicated the tax system is, the tax calculation for a company could be relatively simple, involving a single IF function and a couple arithmetic expressions, or it could involve many nested IF functions if there are several tax brackets to consider.

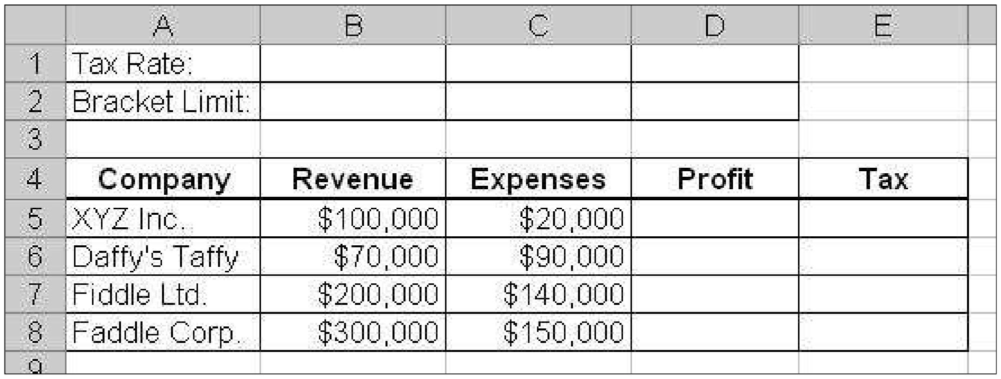
In the case of simple calculations, the tools that you have always used with Excel are sufficient to develop your spreadsheets. However, when you build spreadsheets with complicated calculations Excel provides a powerful tool that will make your spreadsheets easier to read and use. The tradeoff is having to learn how to develop your own customized functions called *user-defined functions* (UDFs). Unfortunately, in order to build a UDF you need to learn how to do some computer programming in a programming language called Visual Basic (VB). Computer programming is the art of using a sequence of statements, that exactly follow a specific structure (or syntax), to ask the computer (or application) to do something for you. In fact, all of the applications you have used on a computer were developed by a computer programmer who was performing computer programming to develop the application. The computer programmer would have followed some computer language.

During lecture you will have learned some of the rules or structure (syntax) for creating acceptable Visual Basic statements.

Exercise: Getting Started

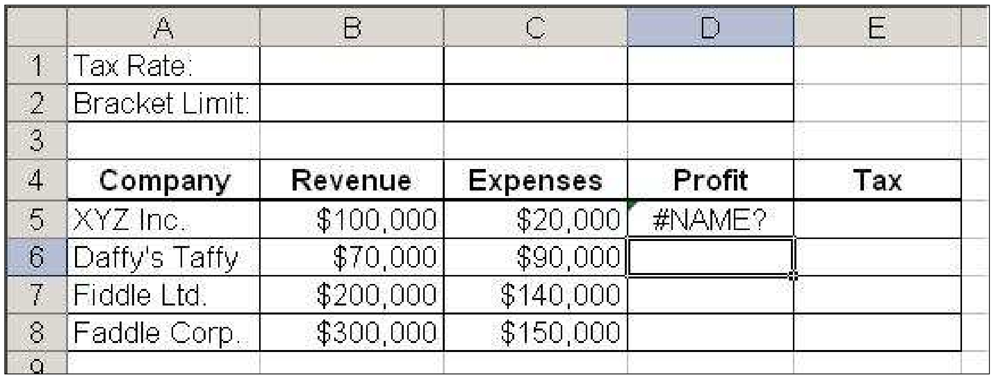
Let's jump right in and create a simple UDF for calculating profit based on revenue and expenses by following the instructions indicated below.

1. Open an Excel workbook and enter the data shown in the table below. Save the file as an **Excel Macro-Enabled Workbook** using *Name*-`Lab1Ex1` as the filename. Use your first name and last initial in place of *Name*. For example, I would save this file as `AlanK-Lab1Ex1.xlsm`.



	A	B	C	D	E
1	Tax Rate:				
2	Bracket Limit:				
3					
4	**Company**	**Revenue**	**Expenses**	**Profit**	**Tax**
5	XYZ Inc.	$100,000	$20,000		
6	Daffy's Taffy	$70,000	$90,000		
7	Fiddle Ltd.	$200,000	$140,000		
8	Faddle Corp.	$300,000	$150,000		
9					

2. Although calculating the profit is a simple calculation that would just involve entering the formula `=B5-C5` into cell D5 and copying the formula down to cell D8, we will instead create a UDF to perform the calculation. The UDF that we will create will be called `PROFIT`. Enter `=PROFIT(B5,C5)` into cell D5 and press enter. You will see this:

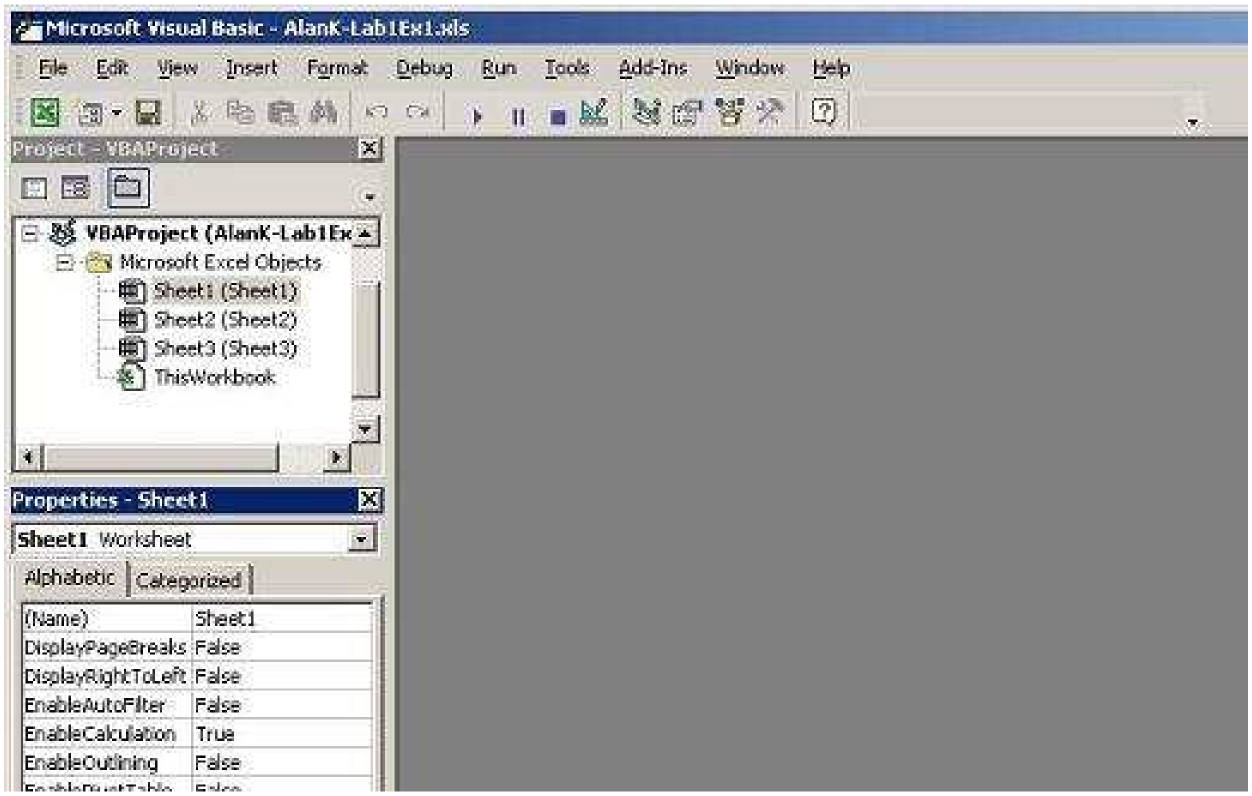


	A	B	C	D	E
1	Tax Rate:				
2	Bracket Limit:				
3					
4	**Company**	**Revenue**	**Expenses**	**Profit**	**Tax**
5	XYZ Inc.	$100,000	$20,000	#NAME?	
6	Daffy's Taffy	$70,000	$90,000		
7	Fiddle Ltd.	$200,000	$140,000		
8	Faddle Corp.	$300,000	$150,000		
9					

3. You will notice that Excel is having some issue with what we just entered into cell D5. Excel accepts what we entered but does not know how to interpret what we typed in. The reason for this is that Excel has no knowledge of a function called `PROFIT` that accepts two values as parameters. Although, this error message is a little unsettling, it should not be unexpected, because we just made up the function name `PROFIT`. We could have just as easily used the function name `BANANA`. But since the UDF that we are about to create is to calculate profit, it makes more sense to call the function `PROFIT`.

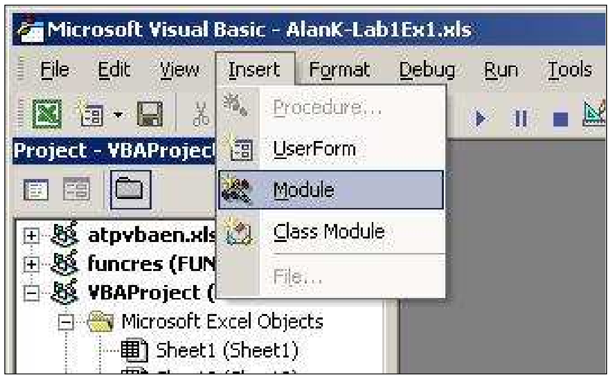
   Now we will start the process of creating the UDF. To do this, we must open a so-called *Visual Basic Module* (VB Module) to be associated with our spreadsheet. To create a VB Module, we need to use the *Visual Basic Editor* (VB Editor).

   Open the VB Editor by pressing **alt-F11** (**alt-F11** is a toggle to move back and forth between your spreadsheet and the VB Editor). The VB Editor window will look something like this:

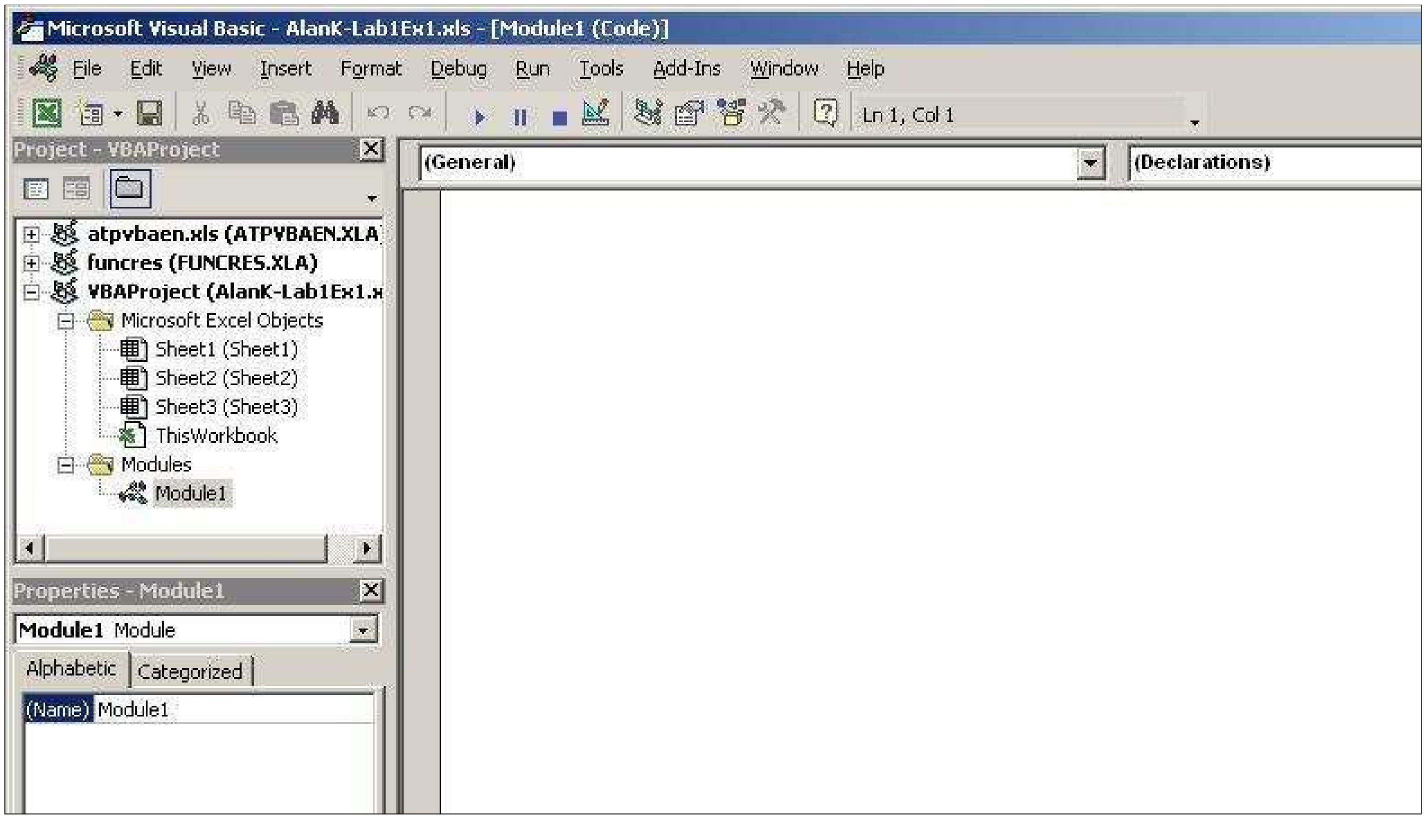


(Name)	Sheet1
DisplayPageBreaks	False
DisplayRightToLeft	False
EnableAutoFilter	False
EnableCalculation	True
EnableOutlining	False

4. Don't be concerned about the complexity of this window. There are only a few buttons and objects with which we will be concerned. To open a VB Module, select **Module** from the **Insert** menu:

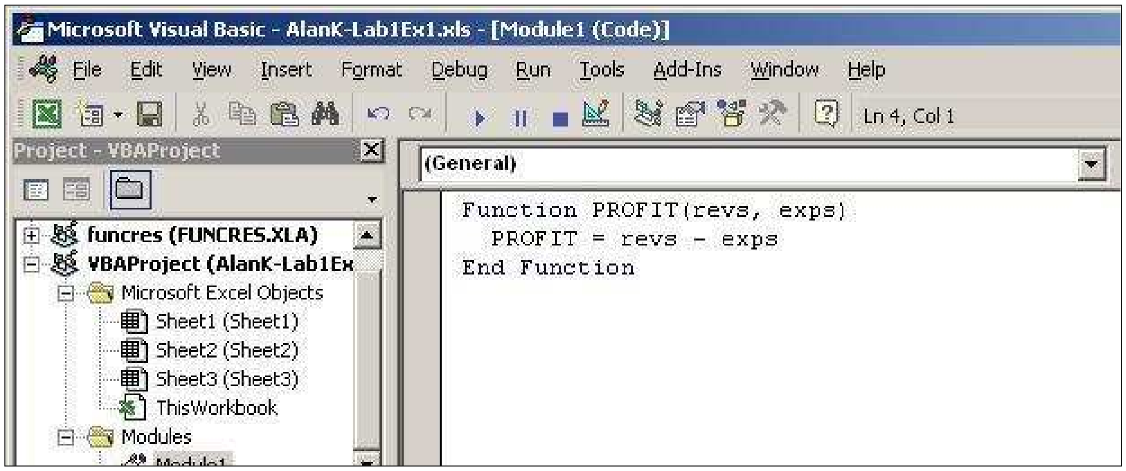


A VB Module will appear in the right side of the window:



It is within this white area on the right that we will create the user-defined function. Notice in the Project window on the left that **Module1** is highlighted. This lets us know that the white area on the right is associated with Module1. Selecting the module can be a little tricky, so be sure that you are comfortable with this (as described during the demonstration).

5. Now that we have a module open, we can *code* the `PROFIT` UDF. Type the VB code for the `PROFIT` function into the VB Module as shown below:



```
Function PROFIT(revs, exps)
  PROFIT = revs - exps
End Function
```

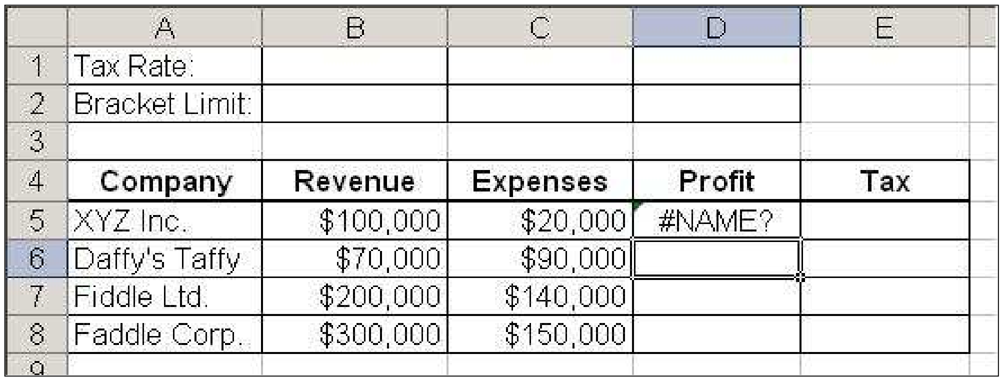
As you type you may make a mistake. Depending on how the mistake was made, one of several things could happen. It will be demonstrated to you what could happen and how you should proceed to fix a mistake.

The syntax to this code was described during lecture, but briefly, here is what each of the words and characters mean:

- The word *Function* is a VB keyword, which is reserved for the purpose of defining a function (i.e. a UDF). The word PROFIT is an *identifier* (or word) that we came up with as the name for the function. This identifier can be any word (group of characters without spaces) that is not already used for some other purpose. For example, the word *Function* is already used by the VB language for another purpose, so we would not be able to call this UDF "`FUNCTION`". We could, if we wanted, have called this function `BANANA` as the word *banana* is not used for anything in VB. However, it is good practice to name UDFs using words that describe what the UDF does. `PROFIT` is more appropriate than `BANANA` in this situation. (I suppose that monkeys might find `BANANA` suitable if that is how they spend their earnings.)
- The first line of this UDF definition is the so-called *function header*. It lets us know how the function can be *called* (or used). The function is called by using the word `PROFIT` followed by a '(', then a value, then a comma, then another value, and finally a ')'.
- The identifiers `revs` and `exps` are variables that are used within the function. Variables in computer programming are similar to variables in mathematics - they represent a value. In computer programming it is more appropriate to say they *store a value*. Because the `revs` and `exps` variables are indicated in the function header within the parentheses, they are called *formal parameters*. Formal parameters are a way for the person using the UDF to pass values into the UDF so that the UDF can perform calculations based on the user's values. Remember that in our spreadsheet we called this UDF from within the formula we created for cell D5 - `=PROFIT(B5,C5)`. Inside the function, `revs` is assigned the value stored in cell B5 and `exps` is assigned the value stored in cell C5. Note that in the spreadsheet formula `=PROFIT(B5,C5)`, `B5` and `C5` are called the *actual parameters*.
- The second line of the function performs an arithmetic operation (namely, subtraction). The value of the variable `exps` is subtracted from the value of the variable `revs`. The resulting value is *assigned* to the variable `PROFIT`. The '=' character in computer programming has a different meaning than it does in mathematics. The '=' character is called the *assignment operator*, and essentially says, take the value calculated on the right of the '=' and *assign* it to the variable on the left of the '='. After the statement executes, the variable on the left *stores* whatever value it was assigned.
- In the second line the variable `PROFIT` is assigned a value. In this context, the identifier `PROFIT` is regarded as the so-called *return value* for the function. Inside a UDF, when you assign a value to a variable that has the same name as the UDF itself, you are actually specifying what value should be *returned* by the function. Remember that a function *returns a value*.
- The third and final line of the function is how VB wants you to indicate the end of the function.

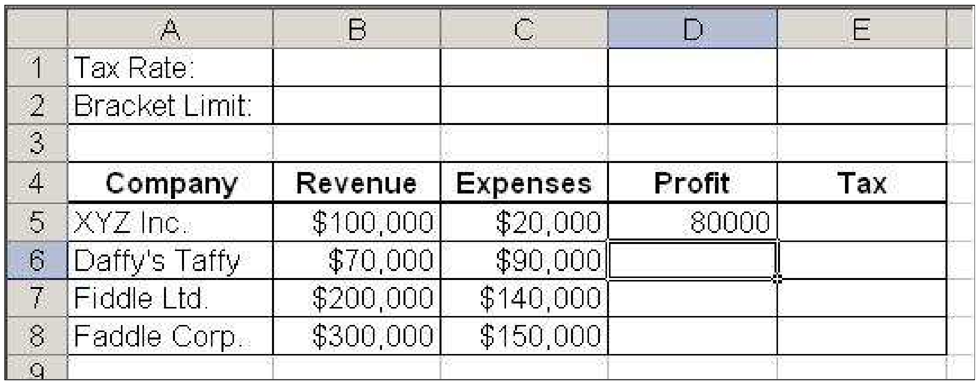
6. Now lets try using our UDF. Recall from Step 2 that when we typed `=PROFIT(B5,C5)` into cell D5, we got the error message #NAME? to indicate that Excel did not recognize a function with the name `PROFIT`. Now that we have specified a function with the name `PROFIT`, namely our UDF, Excel will call upon it to return a value to the formula in cell D5.

   Return to your spreadsheet by selecting it from the taskbar or by pressing **`alt-F11`**. You will still see that cell D5 displays #NAME?:



	A	B	C	D	E
1	Tax Rate:				
2	Bracket Limit:				
3					
4	Company	Revenue	Expenses	Profit	Tax
5	XYZ Inc.	$100,000	$20,000	#NAME?	
6	Daffy's Taffy	$70,000	$90,000		
7	Fiddle Ltd.	$200,000	$140,000		
8	Faddle Corp.	$300,000	$150,000		
9					

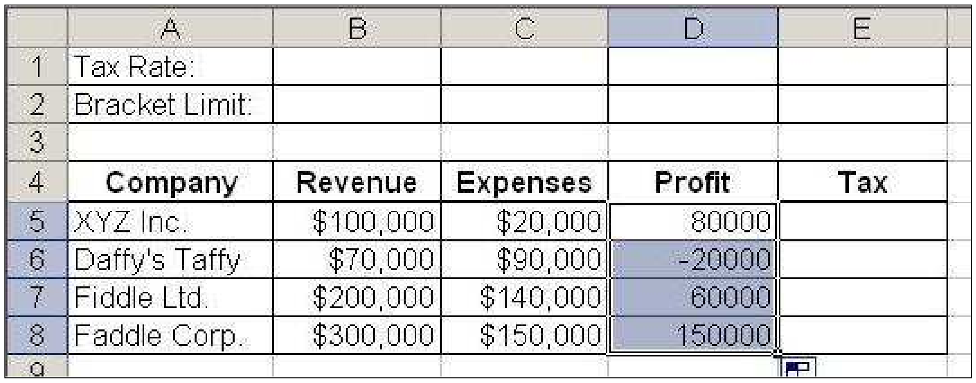
We now just have to tell Excel to recalculate the formulas of the spreadsheet so that it will try the formula in cell D5 again. To ask Excel to recalculate, press **`F9`**. After a moment you will see:



	A	B	C	D	E
1	Tax Rate:				
2	Bracket Limit:				
3					
4	Company	Revenue	Expenses	Profit	Tax
5	XYZ Inc.	$100,000	$20,000	80000	
6	Daffy's Taffy	$70,000	$90,000		
7	Fiddle Ltd.	$200,000	$140,000		
8	Faddle Corp.	$300,000	$150,000		
9					

Since there is now a function (our UDF) that is called `PROFIT`, that matches the way it is used in cell D5 (the word `PROFIT` followed by two values in parentheses) Excel accepts the formula.

7. Copy the formula from D5 to D6:D8:



	A	B	C	D	E
1	Tax Rate:				
2	Bracket Limit:				
3					
4	Company	Revenue	Expenses	Profit	Tax
5	XYZ Inc.	$100,000	$20,000	80000	
6	Daffy's Taffy	$70,000	$90,000	-20000	
7	Fiddle Ltd.	$200,000	$140,000	60000	
8	Faddle Corp.	$300,000	$150,000	150000	
9					

You have now successfully completed the process of creating a UDF and incorporating it into a spreadsheet.

Save your spreadsheet to your network drive or a memory stick and close Excel. You may get a message indicating that you cannot save the file in a macro-free workbook. Follow the instructions and save it to a macro-enabled workbook. Next, try to re-open your Excel file. You may see a message indicating that the spreadsheet contains *macros*, and that this is a potential security threat or a security warning bar may appear above your spreadsheet indicating that macros have been disabled. We will be creating macros in a future lab, but for now you can think of them as small computer programs that exist within a spreadsheet. Macros are created using the VB language in a VB Module, and as a result, our UDF is identified as something that might be a security threat. Generally, you should not enable macros within a spreadsheet with which you are not familiar. You should only enable macros for spreadsheets that you trust and already know contain VB code.

Follow the directions during the lab to adjust the security level so that macros are enabled.

Incorporating VB Structures into A UDF

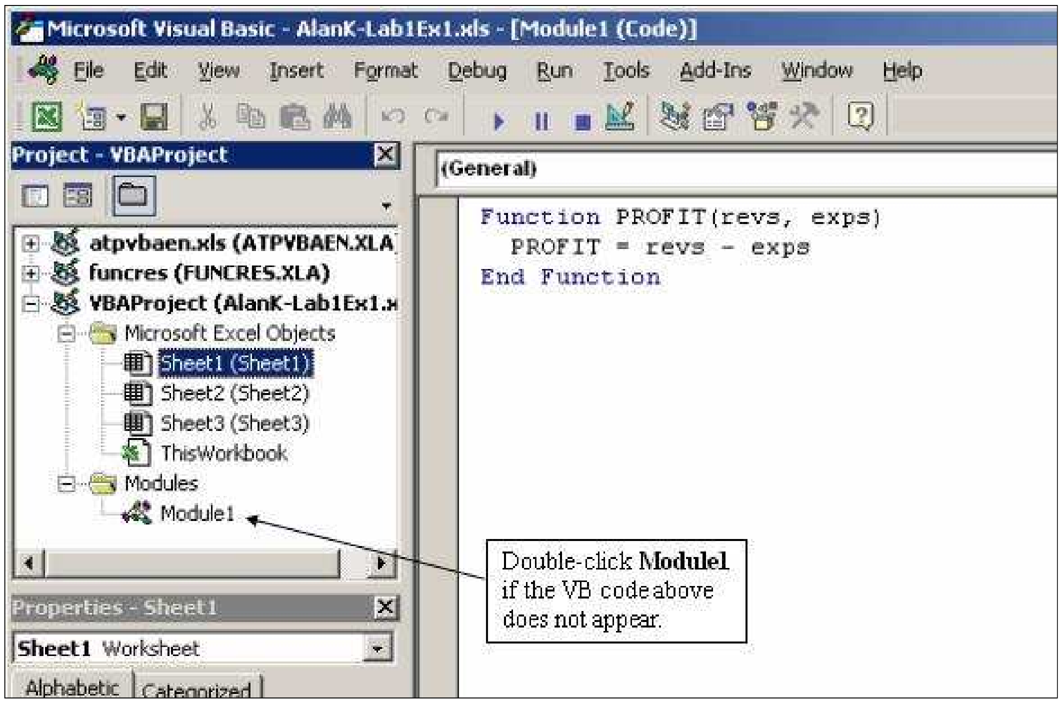
The previous exercise was useful to learn how to create a UDF. However, if that example was typical of the level of problem for which UDFs are created, you would probably never bother with UDFs as it is easier to just perform the calculation in a typical Excel formula. You should consider creating a UDF when the formula for doing so is too complicated to be reasonably incorporated into an Excel formula. Further to this, there are situations where you would find it almost impossible to develop an Excel formula for what you want to calculate. Such a calculation may be relatively simple in a UDF.

Consider the first tax calculations that we performed in class using the built-in IF function. Assuming the tax rate is stored in cell B1 we would use the Excel formula `=IF(D5<0,0,$B$1*D5)` in cell E5. We need an IF function to deal with the situation where we have a negative profit. This formula is not too complicated, but as soon as we start considering different tax rates for different profit ranges, we will need to nest IF functions inside of IF functions making the formula more and more difficult to compose, read and edit. If there were a function that existed to perform the tax calculation based on profit, the formula that we would enter into cell E5 would be simply, `=TAX(D5)`. This would make our spreadsheet easy to read and easy for someone else to understand. However, there is no function called `TAX` that calculates a tax value based on the constraints that we want. So, let's just make one.

Exercise: Creating a TAX Function

To create another UDF, we will return to the VB Module that we were working on previously and append another UDF to the VB Module.

1. Open your workbook from the previous exercise (*Name*`-Lab1Ex1.xlsm`), making sure that macros are enabled.
2. Press **`alt-F11`** to toggle to the VB Editor window. You should see your `PROFIT` UDF created in the previous exercise:

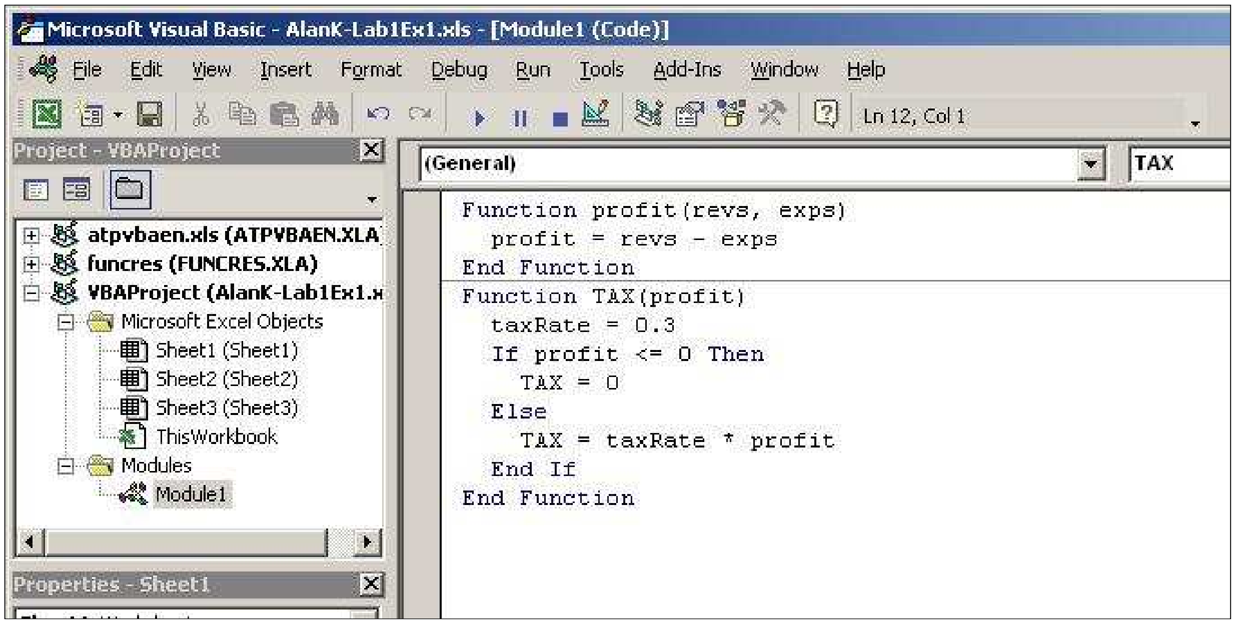


```
Function PROFIT(revs, exps)
  PROFIT = revs - exps
End Function
```

Double-click **Module1** if the VB code above does not appear.

If you just see a blank area on the right, double-click **Module1** in the Project window.

3. Now enter the VB code for our first version of the tax function as shown here:

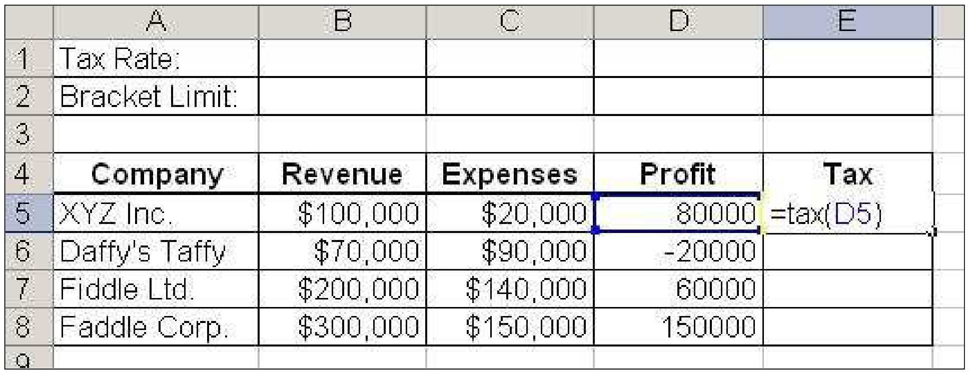

Microsoft Visual Basic - AlanK-Lab1Ex1.xls - [Module1 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 12, Col 1
Project - VBAProject
atpvbaen.xls (ATPVBAEN.XLA
funcres (FUNCRES.XLA)
VBAProject (AlanK-Lab1Ex1.x
Microsoft Excel Objects
Sheet1 (Sheet1)
Sheet2 (Sheet2)
Sheet3 (Sheet3)
ThisWorkbook
Modules
Module1
Properties - Sheet1
(General)
TAX

```
Function profit(revs, exps)
  profit = revs - exps
End Function
Function TAX(profit)
  taxRate = 0.3
  If profit <= 0 Then
    TAX = 0
  Else
    TAX = taxRate * profit
  End If
End Function
```

Upon returning to your workbook, you may find that the UDF does not yet work. Before we fix this, there are a few things that we should note about the `TAX` function that we added to our VB Module.

- As you press enter after typing the first line of the new function, a horizontal line will appear below the `End Function` statement of the `PROFIT` function as a visual aid to distinguish the end of one function from the beginning of another.
- The first line of the `TAX` function follows the same rules as the first line of the `PROFIT` function in that it starts with the keyword `Function` followed by the name of the function, which we arbitrarily chose to be called `TAX`. It includes a single formal parameter which we arbitrarily chose to be called `profit`. Although our choices for function and parameter names were arbitrary, we knew what each was representing so we chose words that were descriptive.
- The second line of the `TAX` function (`taxRate = 0.3`) is an assignment statement that assigns the value 0.3 (i.e. 30%) to the variable `taxRate`. Again, `taxRate` is an arbitrarily chosen name for a variable that we create here to store a value. We use the word `taxRate` for the variable name because it is descriptive of what the variable is storing.
- Lines 3 to 7 of the `TAX` function demonstrate using the Visual Basic *If..Then..Else* statement. It follows rules that are similar to those used for Excel's built-in IF function. The *If..Then..Else* statement follows these rules in the VB language:
  - The first line must follow this syntax: **`If`** *condition* **`Then`** - the keywords `IF` and `Then` surround the specification of a *condition*. Recall that a condition is an expression that results in one of two values - *true* or *false*. Most often a condition is a comparison of two values. In the example above, the value of the variable `PROFIT` is compared with the value 0 using the less than or equal to operator, '<='. The condition expressed by this comparison has the value *true* if the value of `PROFIT` is less than or equal to 0. If `PROFIT` is not less than or equal to 0 the condition has the value *false*.
  - Between the keywords `Then` and `Else`, the programmer puts the sequence of statements that she wants executed if the condition is *true*.
  - Between the keywords `Else` and `End IF`, the programmer puts the sequence of statements that she wants executed if the condition is *false*.
- In the `TAX` UDF we want a tax value of 0 returned if the profit (value stored in the variable `profit`) is less than or equal to 0, so after the keyword `Then` we assign the value 0 to a variable with the same name as the function, `TAX`. Similarly, if the profit is not less than or equal to 0 then we want the UDF to return the product of the profit and the tax rate.

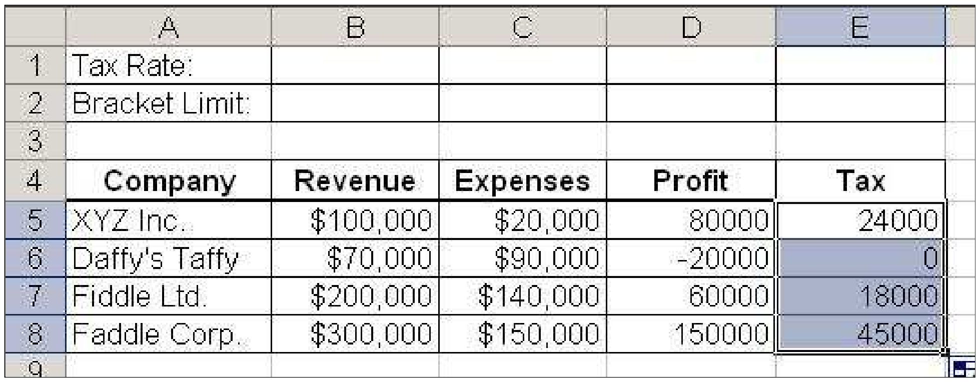
4. Now let's use the `TAX` UDF to calculate the taxes that each company would pay. Return to your spreadsheet (by pressing **`alt-F11`**) and enter the formula shown below into cell E5.



	A	B	C	D	E
1	Tax Rate:				
2	Bracket Limit:				
3					
4	Company	Revenue	Expenses	Profit	Tax
5	XYZ Inc.	$100,000	$20,000	80000	=tax(D5)
6	Daffy's Taffy	$70,000	$90,000	-20000	
7	Fiddle Ltd.	$200,000	$140,000	60000	
8	Faddle Corp.	$300,000	$150,000	150000	
9					

You may find the "#NAME?" message in the cell we just entered the formula into. If this error occurs for you, you will need to adjust the security settings for your file. Select **File**, then **Options** to open the **Excel Options** dialog box. Then select **Trust Center**, then **Trust Center Settings** to open the **Trust Center** dialog box. Then select **Macro Settings** and then *enable all macros* and check *Trust access to the VBA project object model*. After this, save your workbook and close it. When you re-open it, everything should be fine.

5. Press enter and copy E5 to the E6:E8.



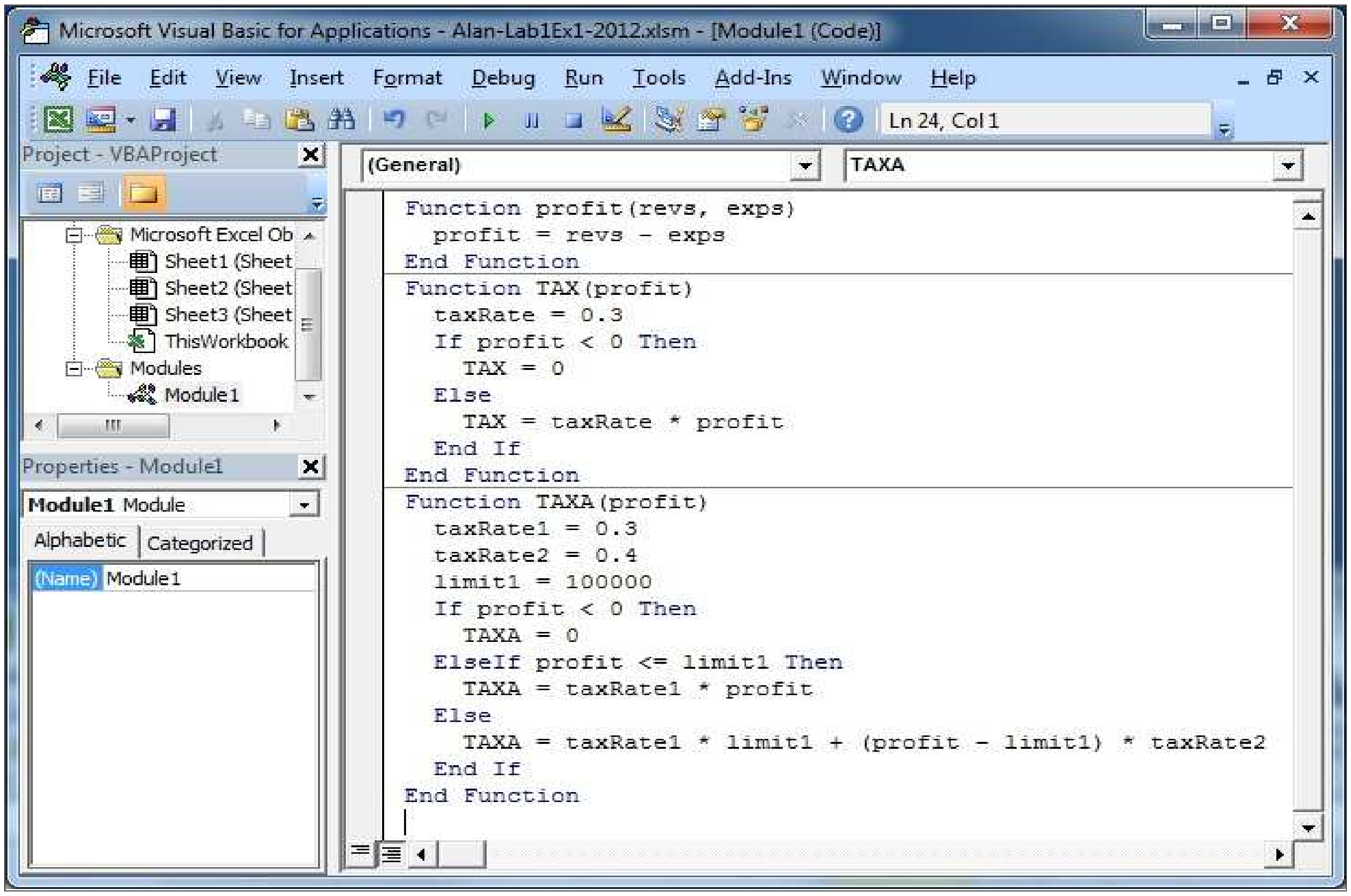
	A	B	C	D	E
1	Tax Rate:				
2	Bracket Limit:				
3					
4	Company	Revenue	Expenses	Profit	Tax
5	XYZ Inc.	$100,000	$20,000	80000	24000
6	Daffy's Taffy	$70,000	$90,000	-20000	0
7	Fiddle Ltd.	$200,000	$140,000	60000	18000
8	Faddle Corp.	$300,000	$150,000	150000	45000
9					

6. Save your workbook.

Exercise: Creating a Complicated TAX Function

In this exercise, we will create a second tax function called `TAX2`. It will look a lot like the `TAX` function we just created, but it will be able to handle two different tax brackets - 30% for profits up to $100,000 and 40% for profits exceeding $100,000.

1. If it is not already open, Open your workbook from the previous exercise (*Name*-`Lab1Ex1.xlsm`), making sure that macros are enabled.
2. Press **`alt-F11`** to toggle to the VB Editor window. You should see your `PROFIT` UDF and your `TAX` UDF. If they are not visible, remember to double-click **Module1**.
3. Enter a third UDF called `TAXA` as shown below.



```
Function profit(revs, exps)
  profit = revs - exps
End Function
Function TAX(profit)
  taxRate = 0.3
  If profit < 0 Then
    TAX = 0
  Else
    TAX = taxRate * profit
  End If
End Function
Function TAXA(profit)
  taxRate1 = 0.3
  taxRate2 = 0.4
  limit1 = 100000
  If profit < 0 Then
    TAXA = 0
  ElseIf profit <= limit1 Then
    TAXA = taxRate1 * profit
  Else
    TAXA = taxRate1 * limit1 + (profit - limit1) * taxRate2
  End If
End Function
```

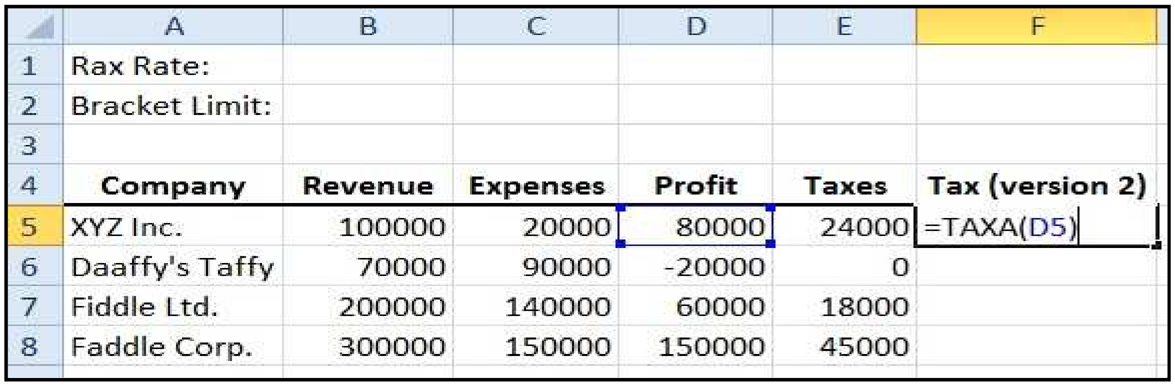
There are a few things that we should note about the `TAXA` function that we just added:

- The area in which you are editing your UDFs has many of the features of a typical editor, so you can cut and paste. Feel free to do so if you think it will help you compose this new UDF.
- The `TAXA` UDF differs from the previous UDF in several ways.
  - There are three variables that are assigned values at the start of the function - `taxRate1`, `taxRate2`, and `limit1`. Each variable is assigned a value that will make our calculations later a little more descriptive and flexible.
  - Instead of an *If..Then..Else* statement we have a slightly different statement, an *If..Then..ElseIf..Else* statement. Actually, this statement is really just a way of putting an If statement inside an *If..Then..Else* statement. We use these types of statements whenever we want to deal with a set of mutually exclusive conditions. In fact, between the first line of the the *If* statement and the *End If* line associated with the *If*, you can include as many *ElseIf* lines as you need to capture all of the conditions you are testing.

    For example, such a syntax might look like this:

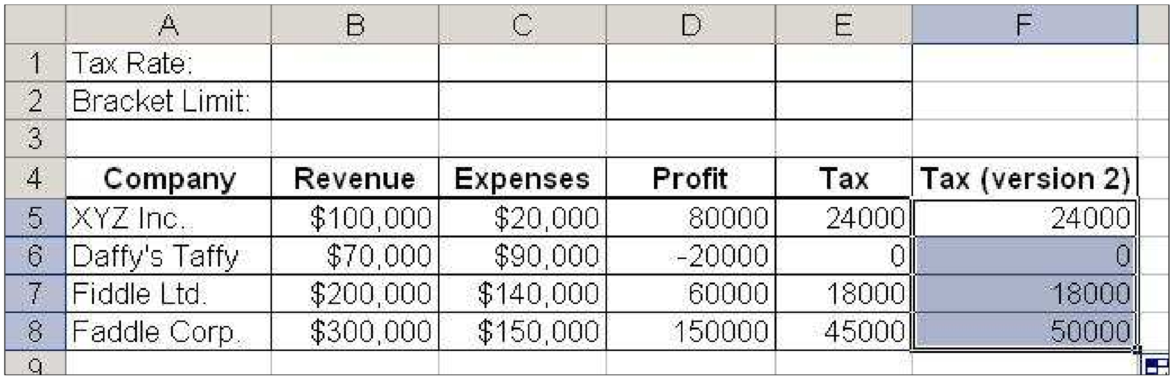
    ```
    If conditionA Then
      statements
    ElseIf conditionB Then
      statements
    ElseIf conditionC Then
      statements
    ElseIf conditionD Then
      statements
    ...
    ElseIf conditionX Then
      statements
    Else
      statements
    End If
    ```

4. Now let's use the `TAXA` UDF to calculate the taxes that each company would pay. Return to your spreadsheet (by pressing **`alt-F11`**) and create a heading for column F and enter the formula shown below into cell F5.



	A	B	C	D	E	F
1	Rax Rate:					
2	Bracket Limit:					
3						
4	Company	Revenue	Expenses	Profit	Taxes	Tax (version 2)
5	XYZ Inc.	100000	20000	80000	24000	=TAXA(D5)
6	Daaffy's Taffy	70000	90000	-20000	0	
7	Fiddle Ltd.	200000	140000	60000	18000	
8	Faddle Corp.	300000	150000	150000	45000	

5. Press enter and copy F5 to the F6:F8.



	A	B	C	D	E	F
1	Tax Rate:					
2	Bracket Limit:					
3						
4	Company	Revenue	Expenses	Profit	Tax	Tax (version 2)
5	XYZ Inc.	$100,000	$20,000	80000	24000	24000
6	Daffy's Taffy	$70,000	$90,000	-20000	0	0
7	Fiddle Ltd.	$200,000	$140,000	60000	18000	18000
8	Faddle Corp.	$300,000	$150,000	150000	45000	50000
9						

6. Save your workbook.

Exercise: Creating a Third TAX Function

Using the techniques of the previous exercises, create a third tax function called `TAXB`, that calculates taxes based on 5 tax ranges as follows:

1. Profits below $0 pay 0 taxes.
2. Profits up to $50,000 pay a tax rate of 30%.
3. Profits above $50,000 up to $100,000 pay a tax rate of 40%.
4. Profits above $100,000 up to $200,000 pay a tax rate of 50%.
5. Profits above $200,000 pay a tax rate of 60%.

Submitting your Work

When you complete these exercises call your instructor over to check your work. If your instructor asks, submit your work by email using Blackboard.