ChEE 201
Computer Reading Number 1

Most students are already very familiar with Excel and many of the tools that are built into the software package. It is a useful package for analyzing data, making plots, and organizing information. However, there is a built-in computer language called Visual Basic (VBA) that gives access to much more powerful data processing tools. This section will introduce students to VBA as it is used in Excel so they can begin to use computer programming to solve problems.

Learning Objectives:

1) Students will learn how to label cells in Excel
2) Students will learn how to send information into Visual Basic from Excel
3) Students will learn how to send information back from Visual Basic to Excel

Before we begin, let's recall that Excel uses a grid of data spaces to hold information. Examine the figure below to get an idea of what this means:

	A	B	C	D	E	F	G
1	Example of Excel Spreadsheet						
2							
3	Equation to be used: $y = mx + b$						
4							
5	Let y be the height of an airplane in feet						
6	Let x be the distance the airplane has travelled since takeoff in feet						
7	Let b be a constant equal to 0.0						
8	Let m be the rate of ascent equal to 10.0						
9							
10	m	10					
11	b	0					
12							
13	x	y					
14	0	0					
15	100	1000					
16	200	2000					
17	300	3000					
18	400	4000					
19	500	5000					
20	600	6000					
21	700	7000					
22	800	8000					
23	900	9000					
24	1000	10000					

We see that we have some text laying out what the sheet will contain near the top of the sheet. We basically have typed in a bunch of information to help us see what the sheet is about in the first 8 lines. In the square A10, we have listed a lable for m, and square B10 has the value for that variable. Likewise, line 11 has the label b in column A and its value in column B. We then have a table of data beginning in line 14 and continuing down that has x values and y values calculated with these constants. This is standard Excel practice and students should be familiar with how to set this up and how to set up the equations to make the table.

For example, students would have put:

$$= \$B\$10 * A14 + \$B\$11$$

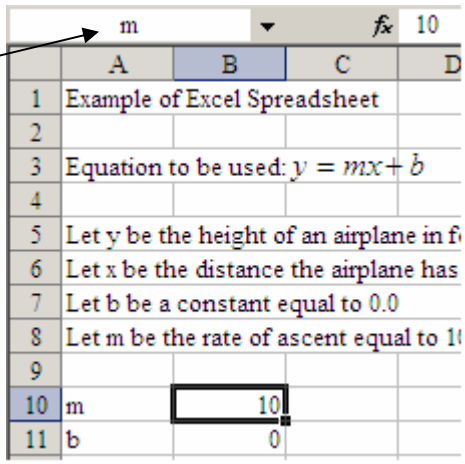
in cell B14. Then they could use Edit Fill Down to copy the formula into the cells below B14. Remember that the dollar signs in the formula protect that cell, and the new formulas beneath the original one will still refer to those cells. Try typing in the formula and doing fill down without the dollar signs to see how the program changes referenced cells if you aren't familiar with this trick.

We will learn a few tools before we begin to set up the Excel VBA. Instead of dollar signs, we could have created a Name for cells B10 and B11. This is done by clicking on:

Insert Name Create Left Column On

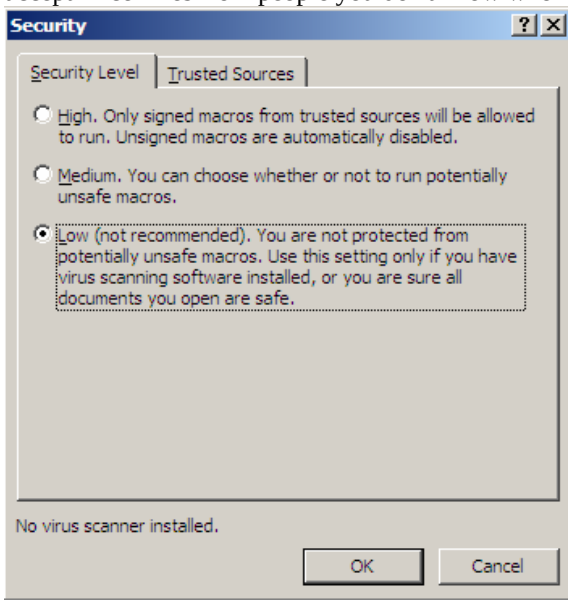
with cells A10 and B10 highlighted. Now, clicking on cell B10, will allow you to see that the value is labelled as m.

Students could now replace B10 in their equation with the letter m and Excel would correctly compute the equations. You can also label b and its value the same way.



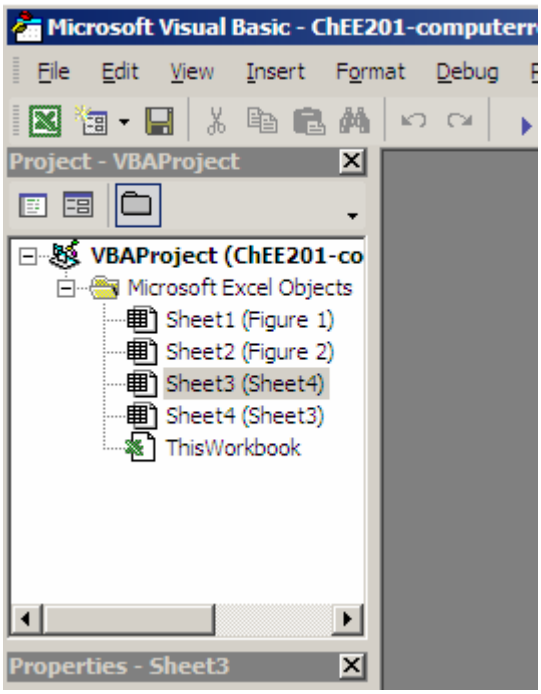
Now that we see how to set up labels, we will turn to how to access VBA in order to send information into and out of the programming language. You can think of Excel and VBA as acquaintances that don't really do a lot together. The analogy is that they are friends that may send each other email back and forth, but they certainly wouldn't go out to dinner together. You'll see what we mean by this in a few more minutes.

The first thing you should do to enable VBA to run on the computer you are working on is to set the security settings. This can be done by going to: Tools Options Security Macro Security or Tools Macro Security depending on which version of Excel you are using and setting this menu to Low. Basically, this will enable Macros to run (and VBA) on the computer you are working on. This security change may enable some Macros to run that are viruses so you should not accept Excel files from people you don't know when your security setting is set to Low like this. The menu you should have just filled out will look like:



When you click OK, you will now be able to run VBA on your computer. If you ever sit down to a new computer and wonder why the VBA program you had been running on other computers is now not working, you should reset the security level to Low on the new computer following the steps we just did.

If we want to get to VBA in Excel, we click on:
Tools Macro Visual Basic Editor
and we get a screen that looks like this in the upper left corner:



Now, click on Insert Module. This opens up a new window where you will type in your computer code so that Excel knows where to find it.

At this point, we need to make a few comments about VBA and how it compares to other programming languages. VBA is great for setting up graphical user interfaces (GUIs) where people click on menus and enter information into boxes. If you've taken a JAVA class, you may already be familiar with these tools and how to use them. However, VBA also can be used much as one uses C, C+, FORTRAN, BASIC, or any of the other computer languages you may already be familiar with.

While students are often wary of learning computer programming, they soon realize that once they have learned how to program in one language, then other

languages are easy to pick up and use. The differences between computer programming languages and how you get them to operate is in the syntax you use in order to get the program to understand what you want it to do. A good source of information about any language is the internet. If you ever find yourself stuck and you aren't sure how to create a short section of code to do something, try using a search engine and typing in, for example: Visual Basic AND Tutorial. You'll be surprised at how much free information is out there on how to program.

Let's now see how to send information into and out of VBA. Type the following lines into your open VBA module window that came up when you did Insert Module:

```
Option Explicit
Function Send(m)

Send = m
End Function
```

Notice that the program automatically added a line under Option Explicit and it also automatically added End Function once you typed in Function. The Option Explicit tells the VBA program that you will be defining and dimensioning the variables much like you would in FORTRAN. This allows you to keep track of your variables and makes sure you aren't making any mistakes. If this doesn't seem to make sense to you yet, basically, we will need to tell VBA at the beginning of our program what our variables are and what type they are (integers, decimals, strings of characters, etc.) We'll come back to this topic in a little while so don't worry if it seems confusing right now.

Now that you have set up this very short program, go back to your Excel sheet and enter "=Send(B14)" into cell C14. Now copy this formula (C14 cell) into the cells beneath C14. What do you notice?

We see that we are just getting back the number that is in the column to the left of column C. Let's look at our program now and figure out what it was doing. If you remember, we said that VBA and Excel are acquaintances but not really good friends. When you typed "=Send(B14)" into Excel, you were telling Excel to package up the information from cell B14 and send it to its acquaintance, VBA. VBA will just sit there patiently waiting until it receives this information before it will start to run the program. Once it receives the information in Send, it runs through the program. When VBA encounters "Send = m" at the end of the program, it takes m, packages it up and send it back to its acquaintance Excel to put in the cell where you put in your formula.

One thing you should notice is that you have used predefined functions in Excel all the time in the past. You are probably used to entering something like "=exp(A14)" into a cell to take the exponential value of the number in cell A14. Well, with VBA, we have just created our own function called Send. This brings up one point that you must be careful about when creating a VBA programming function. You aren't allowed to create a function that has a name that is the same as one already predefined in VBA or Excel. If you used Exp instead of Send, the program might either give you the exponential of the value or it might give you an error. Since there are thousands of functions predefined in Excel, you may need to try a few names for your functions before you find one that works.

Let's try a few things here to have you become a little more comfortable with how VBA and Excel are working. Let's add a few lines to our VBA module until it looks like this:

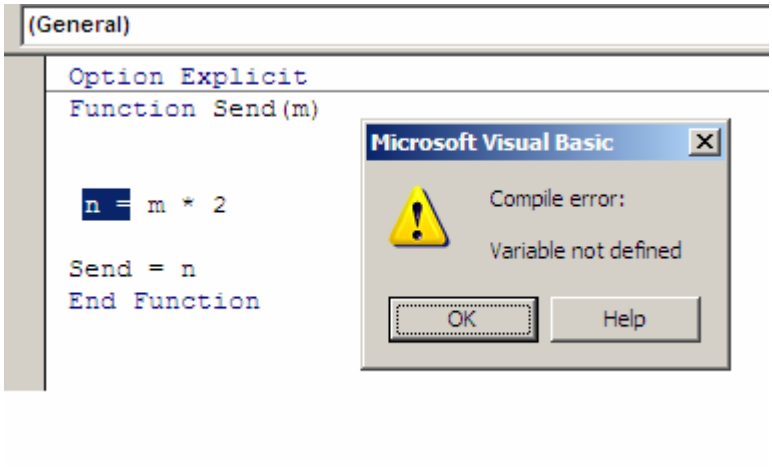
```
Option Explicit
Function Send(m)
Dim n as single

n=m*2

Send = n
End Function
```

When we make these changes to our program, we see that we are still sending in information from the cell to the left of the one we are typing "Send" into. However, the VBA program now adds a new variable, n, to the ones it can use in order to do calculations. We dimensioned this variable with the line "Dim n as single", telling it we have a new variable, n, and that we will be using single precision. We'll come back to this issue of how dimensioning numbers can affect your answers in a later reading. For now, there are two good choices for the numbers we will be handling, which are the one we just used and "Dim n as double". The difference between the two is that double carries twice as many spaces to hold numbers as single does so that you may get more accurate answers.

The middle line of the program "n = m*2" takes the value m, multiplies it by 2 and then sends that information back to n. n is now equal to the value m that came into the program (cell A14 for the top line) times 2. Send = n then takes the value of n and packages it up so Excel can put the answer in the cell.

Learning errors and where they come from:
A good practice whenever you are learning how to use a computer program is to see what errors you get when you knowingly do something wrong. Try the following and write down what happens when you do it:
First, remove "Dim n as single" from the last program you were just working on and go back into Excel and click on cell A14 and hit enter.

You'll see that you get an error like the one to the right. VBA is a fairly nice programming language to use in some respects because it often tells you when you've made an error. Here, it gives us the window telling us what it thinks is wrong. Then it highlights in the program where it encountered the error.



VBA also won't let us continue until we acknowledge the error and try to fix it. On the VBA menu, you'll now have to click on OK and add the dimensioning line back in. Once you have done that, you'll need to click on Run Reset in order to allow Excel and VBA to talk to one another again. If you looked back at the Excel sheet in the middle of testing this error, you would have seen something like #VALUE! or #NAME? in some of the cells. We'll come back to this phenomena in a little bit since these are other indications that your program isn't doing what it's supposed to be doing even if VBA isn't giving you an error. One example is if you haven't reset your security settings to low. Then Excel won't be able to get information to VBA and you'll probably get the #NAME? result in your cell where you entered your formula.

In this case where we've just put the dimension statement back in, we see that the correct answers now come back into our cells once we've done Run Reset in the VBA module.

Let's try a few other ways of getting errors now that you see what happens.
1) Change Send=n to Send=unknown and see what happens
2) with Send=unknown there, change the dim statement to "dim n as single, unknown as single" and click on Run Reset. Go back to Excel and see what you get as a result.
3) Start over with your working program and change "Function Send(m)"to "Function Send(p)". What does Excel give you and what error do you get?

Let's look at a slightly different program to do some other calculations. Close out of Excel and VBA to begin over with a new example that takes two numbers into VBA and computes y from them, just like Excel did in our original figure. Type the following into your new VBA module:

```
Option Explicit
Function line(m, x, b)
Dim y As Single
y = m * x + b
line = y

End Function
```

Now, set up a table like was shown in the first figure of this section. Label the m and b cells. Now, go to cell A 14 and enter "=line(m,A14,b)". With our two labels for m and b, we get the same answers in the first figure. If we copy the cell down into the places beneath it, we see that the A14 reference changes and we get all the answers we expect.

We now have seen how to send more than one variable into VBA and how to use the numbers to calculate other values. This will be useful later on as we move to more complicated programs.

You are now ready to do Computer Homework 1.