ChEE 201
Computer Reading Number 2

Now that you have been through Computer Reading 1 and have done HW 1 on that material, you are ready to build upon your programming knowledge and learn how to do more powerful activities in VBA. The program format that we learned in Reading Number 1 will be used for almost all of our VBA programs so we'll keep coming back to that format over and over again throughout the computer assignments. We'll only change our program structure from the type you saw when we encounter programs that need to send in a bunch of numbers or need to output more than one number, much like when we encounter gaussian elimination as a method of solving systems of linear equations. This will come back in a later reading.

Learning Objectives:
1) Students will learn how to use If…Then…statements to make decisions so that programming logic will become useful
2) Students will learn how to use loops to do repetative actions so their code will run faster and look better
3) Students will learn how to use Message Boxes in VBA to check that their programs are doing what they are supposed to be doing and making it through the code

Students are expected to have worked through Reading 1 and the related HW assignment. If they have not, they will be sent to do these activities before help will be given on this section of reading.

If…Then…Statements:
Computer programming like you did on the first reading wasn't very powerful. It's much more useful to be able to make decisions about variables and what to do when choices can be made. The difficulty is in getting the logic right so that the decisions lead to outcomes that make sense and match what you wanted to do. Getting the decisions right is very important because the computer code will only do exactly what you tell it to do. Let's think of an analogy to see what this means.

If you have a niece, nephew, child, or have ever baby sat someone who is around 5 years old, you have already done "programming." What happens when you tell the child to clean their room? Nothing? Or maybe you look at the room later on and you find that nothing is even close to what you thought cleaning the room meant it should be; their clothes are hidden under the bed, their books are piled in the bottom of the closet, and their shoes are hanging from the doorknob. If you want a 5 year old to do something, you have to carefully give your list of instructions on what cleaning their room entails. You would have to give a program like:

Open your closet doors
Pick up both of your shoes
Put both shoes on the floor in your closet lined up on the right side
Close your closet doors

If you have any books on the floor, pick them up
    Put the books you have on the book shelf

If you have any clothes lying around, pick them up
    Put the clothes you have in the wash hamper in the bathroom

End cleaning your room

Computer programs are the same way for needing explicit instructions. Whatever you tell them to do may get done if you are specific and correct enough in your logic and explanation.

Let's look at an example using If…Then…statements and decision trees to reach a computer result so that you can get a better idea of how to program this specificity in VBA. Open a new Excel file and set up the following VBA module:

Option explicit

```
Function iffy(a,b)
Dim m as single

If a<b then
        m = b
Else
        m = a
End if

iffy = m
End Function
```

What do you think this program will do?

Well, it takes in two numbers sent to VBA in the calling statement and only sends back the highest number of the two to the cell where you enter "=iffy(num1, num2)". Try this in Excel to see how it works.

If you are only doing one check and there isn't anything else you want the program to do, you can use:

If (conditional statement) Then (what happens)

without using End If or Else. An example would be if you wanted to see if a value was zero or not and then set another variable based on this answer. All you would need would be:

If value=0 Then flag = 1

Setting up longer decision trees is a little more difficult, but you'll get some practice on this homework assignment. Seek help from the instructor if you are having difficulty with the program for that part.

Loops:

Computer programming loops are very useful when you have to do a certain operation over and over again. Let's say that you wanted to add up all the numbers from 1 until 10 and then report the sum. A program that would do this would be:

```
Option Explicit
Function Loopy(n)
Dim I as single, sum as single

sum = 0

for I = 1 to n
        sum = sum + I
next I

loopy = sum
End Function
```

To complete your VBA/Excel program, you should enter "=loopy(10)" into cell A1. Your result in that cell should be 55. Check that this is true. Now, on your calculator or by hand, do the sum of:

1+2+3+4+5+6+7+8+9+10 = ?

The result should be 55.

A few comments should be made about this program so you can see some of the details discussed in this and Reading 1. Try changing loopy to loop. What happens?

You should see that the lines with loop become red. This is because loop is a built in function and we've tried to create another function with the same name as a built in one. This is why we called it loopy instead of loop.

We have to start our loop by initializing the variable "sum" to being zero. If we don't do this, your computer could have stored a value for "sum" in a previous calculation that you don't know about. If you don't reset it to zero, the program this time around could just start with it already being equal to the value that you last had for sum. An analogy will help explain this:

Imagine I wanted to give you directions to my house and I said:

"Go east until you reach Norris. Turn left. Go one block and turn right. Go two houses and turn left"

This is very different from:

"**Starting at my office**, go east until you reach Norris. Turn left. Go one block and turn right. Go two houses and turn left"

The second set of instructions tells you where to start and what to do. You probably wouldn't have found the right destination if you followed the instructions in the first case from wherever you happened to be when you started.

The next information to notice about loops is that you have a variable that is incrementing (or decrementing), I, in this case, and it starts at a certain number and ends at another one. You could also set it to use only to even numbers by replacing the current line with:

For I = 1 to n step 2

This will go 1, 3, 5…all the way to n. Try it and see what your result is. What did you think it would be if you did it by hand? (You should have gotten 25…why wouldn't the last number, 10, be counted?) You can change the step size to be anything, small, large, or even negative, depending on what you are trying to do with your loop.

You need to dimension your looping variable names if you are using Option Explicit because the program won't know what to do when it reaches the For…Next…line if you don't. Remember to always do this. This is why we added Dim I as Single.

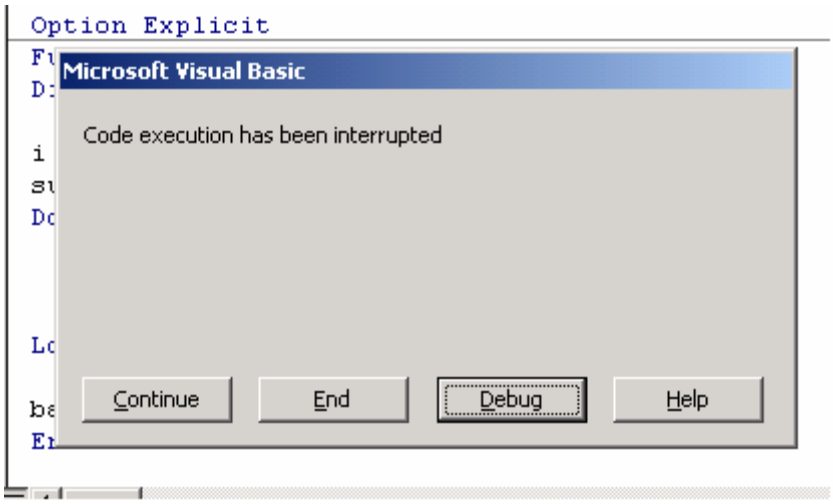
While there are other looping structures you could use in VBA, you can end up with problems if you use them. An example is the popular Do…While…Loop. This basically tells the computer to keep doing an activity until it passes some point. An example would be:

```
Option Explicit
Function badloop(n)
Dim sum As Single, i As Single

i = 0
sum = 0
Do
   i = i + 1
   sum = sum + i
   If i = n Then Exit Do
Loop

badloop = sum
End Function
```

This works and gives us the right answer…So why would we choose to avoid this kind of logical construction? Let's try changing the line with "i=i+1" to "i=i-1". What happens when you run the program? The Excel program locks up and you can't seem to do anything to stop it. Why is this? It's because you have created a condition that will never be met in the Do loop so it will keep going forever. The best way to get your program to stop is to hit Ctrl+Pause/Break on your keyboard to get to a screen that looks like:



We see here that we need to click on "End" to get our program to be stopped. This is why you should avoid Do…while loops if you can.

This example may appear trivial. You would recognize as you programmed this short set of code that you were setting up your condition for failure. When you get to longer programs with many more variables that might all be interdependent, it is much more difficult to spot this error. So, always use the For….next…loop structure if you can. We'll do more with loops over time and you'll become more comfortable with them.

Message Boxes:

While we saw in Reading 1 that VBA often indicates where you've made an error by stopping the code and highlighting the area where the problem was encountered. But this can't be counted on if you have a logical error or if the error is subtle in some way. Instead, we can turn to message boxes that will show us what is happening in our program as it occurs instead of waiting until the end of the program run to see. Try the following very simple program in a VBA module:

```
Option Explicit
Function Looped(n)
Dim i As Single, sum As Single

sum = 0

For i = 1 To n Step 1
   sum = sum + i
   MsgBox (i)
Next i

Looped = sum
End Function
```

Put "=looped(10)" in cell A1 and hit enter. Why did we call the program function Looped instead of Loop? Go back to the earlier part of this reading to see why. Also, we could have used Loopy like we did earlier as long as we closed out of Excel and started with a new file where we were working. Each module we create is linked to all the other Excel sheets and modules we have open on the computer as we have been using it so changing variable names is one way to keep working with a single sheet.

What happened when you hit enter? You should have gotten a window that said "1" OK? If you click OK, it will say "2" OK?…etc. This goes until i hits n. (Again, Ctrl + Pause/Break") can get your program to stop running if you have the message box in a place that makes it keep popping up unintentionally and you want to get out of that loop. So, you see that the message box is a great way to get an intermediate number, I in this case, out of your program so you can see what your program is actually doing. Keep this in mind as you program more complex code later on.

More about message boxes and how to use them is available in online tutorials you can find using internet search engines so little more will be discussed here. At this point, you know enough to attempt the second computer homework.