AN INTRODUCTION TO REPEATED OPERATIONS IN MATHEMATICA

We are required, very commonly in scientific programming, to perform series of sequential calculations. The need to evaluate a function or expression repeatedly requires constructing a programming loop. There are several ways to do this in Mathematica; in this classnote, we learn the basics of programming loops. Please see the doc center section "Loops and Control Structures" for more details and many examples.

The "Do" Statement :

Suppose we want to find the sum of the first 100 integers; we know we can do this simply using the Sum command :

```
In[329]:= Sum[k, {k, 1, 100}]

Out[329]= 5050
```

While this is the best way to do such a computation in Mathematica, we can illustrate the structure of various loops by performing the sum calculation several different ways. For instance, using the Do command :

```
In[367]:= Clear[sum, n]
          sum = 0;
          Do[sum = sum + n, {n, 1, 100}]
          Print[sum]

5050
```

In this program, we continually update the value of sum by adding its previous value to the current value of the index n. If we wished to add only the odd numbers, we write :

```
In[374]:= Clear[sum, n]
          sum = 0;
          Do[sum = sum + n, {n, 1, 100, 2}]
          Print[sum]

2500
```

The "While" Statement :

The same calculation can be done in a "While" statement. In a "While" statement, the evaluation is iterated as long as the test statement remains true. In our case we would write :

In[381]:=
```
Clear[sum, k]
sum = 0; k = 1; While[k < 101, sum = sum + k; k++]
Print[sum]
```
```
5050
```

Look at this statement carefully. What do the semi - colons do? What do you think k++ means? What would happen if we did not include the k++ statement?

The "For" Statement

The "For" statement requires that we inlude a starting point, a test condition, an increment, and the statement we are to execute or evaluate. In our case, our statement looks like :

In[420]:=
```
Clear[sum, k]
sum = 0;
For[k = 1, k < 101, k++, sum = sum + k]
Print["sum of first ", k - 1, " integers = ",
 sum, "   ", "number of terms evaluated = ", k - 1]
```
```
sum of first 100 integers = 5050   number of terms evaluated = 100
```

This output allows us to see that the test (in this case, testing whether the index k is less than 101) is done before the evaluation. The final iteration of the program indexed the value of k to 101; when this value of k was tested in the While statement, the execution ended and the loop was exited since k no longer satisfied the test that k < 101. Therefore, the final evaluation of sum occurred when k = 100; once k = 101, the loop terminated before k = 101 could be added to the sum.

One final example :

We have spent some time studying the Levi - Civita permutation tensor this term. A useful Mathematica function to represent this tensor is the Signature function :

In[426]:=
```
Signature[{1, 2, 3}]
```
Out[426]=
```
1
```

In[427]:=
```
Signature[{1, 3, 2}]
```

Out[427]=
```
-1
```

In[428]:=
```
Signature[{1, 1, 2}]
```

Out[428]=
```
0
```

I think you can figure out how this works; the doc center will provide more details as needed.

Now, suppose we want to print out all 27 possible values of ϵ_{ijk}. We can write:

In[429]:=
```
Do[Print[Signature[{i, j, k}]], {i, 3}, {j, 3}, {k, 3}]
```

```
0
0
0
0
0
1
0
-1
0
0
0
-1
0
0
0
1
0
0
0
1
0
-1
0
0
0
0
0
```

Fascinating. But this shows how you can do a do loop over three repeated indices. When we learn "If" statements shortly, we will see how we could easily change this code to print only non - zero statements.