In high school you may have learned one way to multiply matrices.
Chatper 2.4 of your text illustrates many ways.

1. The high school or dot product way appears on pages 67 and pages 68.
   It says that the entry in the (i,j) position is the sum over k from 1 to n of a[i,k]b[k,j].

2. The matrix times column away appears on the top of page 69.

3. The row times matrix approach, also on the top of page 69.

4. The column times row approach in Example 3 on the top of page 71.

Part 1. The code below implements a basic matrix multiply for square matrices.
If you are not using Julia, write one in your language. Either way check a few 3x3 examples, say, and see that it is correct.

In [1]:
```
function matmul_ijk(a,b)
    n=size(a,1)
    c=zeros(a)
    for i=1:n, j=1:n, k=1:n
        c[i,j] +=  a[i,k] * b[k,j]
    end
    c

end
```

In [2]:
```
A=rand(0:2,3,3); println(A);
B=rand(0:1,3,3); println(B);
matmul_ijk(A,B)
```

```
[1 1 0
 0 0 2
 1 1 0]
[1 1 0
 0 0 0
 1 1 1]
```

In the code above the "i loop" is the outer loop. It runs the slowest, like the leftmost odomoter digit in a car. The "j loop"is next, and the "k loop" runs the fastest like the rightmost odomoter digit in a car.

Part 2: There are six ways to reorder these three loops. Are they all correct matrix multiplies? For example try the "jki" method:

In [3]:
```
function matmul_jki(a,b)
    n=size(a,1)
    c=zeros(a)

    for j=1:n, k=1:n, i=1:n
        c[i,j] +=  a[i,k] * b[k,j]
    end
    c
end
```

Out[3]: matmul_jki (generic function with 1 method)

In [4]:
```
matmul_jki(A,B)
```

Part 3: There are six ways to reorder the loops ijk,ikj,jik,jki,kij,kji.
For each of the six ways, decide whether it is of type 1, type 2, type 3, or type 4 in the list repeated here for convenience:

When there is more than 1 of a particular type, explain how they are different in terms of the order in which elements are computed.

Make a nice table

Some codes that follow may help in your investigation, and might even be fun to play with, but the problem set ends with this Part 3.

1. The high school or dot product way appears on pages 67 and pages 68.

2. The matrix times column away appears on the top of page 69.

3. The row times matrix approach, also on the top of page 69.

4. The column times row approach in Example 3 on the top of page 71.

Print every step of the matmul algorithm:

In [5]:
```
function matmul_ijk(a,b)
    step=0
    n=size(a,1)
    c=zeros(a)
    for i=1:n, j=1:n, k=1:n
        c[i,j] +=  a[i,k] * b[k,j]
        step+=1
        println("Step ", step," :","\n",c)
    end
    c
end
```

Out[5]: matmul_ijk (generic function with 1 method)

o=int(ones(3,3)) # You can use any matrix, but I think the integer matrix of ones makes it easy matmul_ijk(o,o);

In Julia and perhaps your language, there are knobs that make it even easier to watch the algorithm in action.

In [6]:
```
using Interact # Only needed once to load @manipulate
```

Add code to stop at a certain step

In [7]:
```
function matmul_ijk(a,b,stop)
    step=0
    n=size(a,1)
    c=zeros(a)
    for i=1:n, j=1:n, k=1:n
        if step==stop;  return(c); end
        c[i,j] +=  a[i,k] * b[k,j]
        step+=1
    end
    c
 end
```

In [8]:
```
n=10
o=int(ones(n,n))
@manipulate for stop=0:n^3
    matmul_ijk(o,o,stop)
end
```

Out[8]:
```
10x10 Array{Int64,2}:
 10  10  10  10  10  10  10  10  10  10
 10  10  10  10  10  10  10  10  10  10
 10  10  10  10  10  10  10  10  10  10
 10  10  10  10  10  10  10  10  10  10
 10  10  10  10  10  10  10  10  10  10
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
```

In [9]:
```
function matmul_kji(a,b,stop)
    step=0
    n=size(a,1)
    c=zeros(a)
    for k=1:n, j=1:n, i=1:n
        if step==stop;  return(c); end
        c[i,j] +=  a[i,k] * b[k,j]
        step+=1
    end
    c
 end
```

In [10]:
```
n=10
o=int(ones(n,n))
@manipulate for stop=0:n^3
    matmul_kji(o,o,stop)
end
```

Out[10]:
```
10x10 Array{Int64,2}:
 5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5
```