PS1 Solution Template

a. Using the matrix squaring operator create a "triangular" matrix with 1 on the main diagonal, 2 above, etc.

$$M(n) = \begin{pmatrix} 1 & 2 & \dots & n-1 & n \\ & 1 & 2 & \dots & n-1 \\ & & \ddots & \ddots & \\ & & & 1 & 2 \\ & & & & 1 \end{pmatrix}$$

In[1]:
```
# The line below is a Julia function that given n computes M(n)
# If you are not using Julia please supply your function
M(n) = triu(ones(n,n))^2
# Show that it works
M(5)
```

Out[1]:
```
# Put your output here

5x5 Array{Float64,2}:
 1.0  2.0  3.0  4.0  5.0
 0.0  1.0  2.0  3.0  4.0
 0.0  0.0  1.0  2.0  3.0
 0.0  0.0  0.0  1.0  2.0
 0.0  0.0  0.0  0.0  1.0
```

In[2]:
```
# Show that it works for n=1,2,3,4,5,6
{M(n) for n=1:6}
```

Out[2]:
```
# Put your output here

6-element Array{Any,1}:
 1x1 Array{Float64,2}:
 1.0
 2x2 Array{Float64,2}:
 1.0  2.0
 0.0  1.0
 3x3 Array{Float64,2}:
 1.0  2.0  3.0
 0.0  1.0  2.0
 0.0  0.0  1.0
 4x4 Array{Float64,2}:
 1.0  2.0  3.0  4.0
 0.0  1.0  2.0  3.0
 0.0  0.0  1.0  2.0
 0.0  0.0  0.0  1.0
 5x5 Array{Float64,2}:
 1.0  2.0  3.0  4.0  5.0
 0.0  1.0  2.0  3.0  4.0
 0.0  0.0  1.0  2.0  3.0
 0.0  0.0  0.0  1.0  2.0
 0.0  0.0  0.0  0.0  1.0
 6x6 Array{Float64,2}:
 1.0  2.0  3.0  4.0  5.0  6.0
 0.0  1.0  2.0  3.0  4.0  5.0
 0.0  0.0  1.0  2.0  3.0  4.0
 0.0  0.0  0.0  1.0  2.0  3.0
 0.0  0.0  0.0  0.0  1.0  2.0
 0.0  0.0  0.0  0.0  0.0  1.0
```

b. You very likely have heard of the triangular numbers: (see wikipedia if not)

$$T_n = 1 + 2 + \ldots + n.$$

In[3]:

```
#Here they are
cumsum(1:10)'
# If not using Julia, how would you do this in your language?
```

Out[3]:

```
1x10 Array{Int64,2}:
 1  3  6  10  15  21  28  36  45  55
```

Don't use cumsum, or sum or "+", just matrix operations to create the matrix that has the triangular numbers on the diagonals: Explain roughly (not too formal a proof), why your idea works.

$$M(n) = \begin{pmatrix} 1 & 3 & \ldots & (n-1)n/2 & n(n+1)/2 \\ & 1 & 3 & \ldots & (n-1)n/2 \\ & & \ddots & \ddots & \\ & & & 1 & 3 \\ & & & & 1 \end{pmatrix}$$

In[4]:

```
# Put your input code here, put in a function and show that it
works, just like In[1] and Out[1]

M(n) = triu(ones(n,n))^3
M(5)
```

Out[4]:

```
5x5 Array{Float64,2}:
 1.0  3.0  6.0  10.0  15.0
 0.0  1.0  3.0   6.0  10.0
 0.0  0.0  1.0   3.0   6.0
 0.0  0.0  0.0   1.0   3.0
 0.0  0.0  0.0   0.0   1.0
```

c. Don't stop. Keep going, and get the tetrahedral numbers. (see wikipedia) Explain briefly why this worked.

In[5]:
```
# This is hardly much different from In[4] and Out[4]

M(n) = triu(ones(n,n))^4
M(5)
```

Out[5]:
```
5x5 Array{Float64,2}:
 1.0  4.0  10.0  20.0  35.0
 0.0  1.0   4.0  10.0  20.0
 0.0  0.0   1.0   4.0  10.0
 0.0  0.0   0.0   1.0   4.0
 0.0  0.0   0.0   0.0   1.0
```

If you think about how matrix multiply works, if you multiply any matrix A by `triu(ones(n,n))` the (i,j) element of the result is the partial sum of the ith row up to the jth column. Said more simply, it's like doing a `cumsum` of each row. That's exactly how the tetrahedral numbers are defined, as the cumsum of the triangular numbers.

d. Let $A = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$ and $B = \begin{bmatrix} -1 & 2 & 1 & 4 \end{bmatrix}$.

Compute $(AB)^{10}$ on your computer. Explain why it's possible to get this without a computer!

In[6]:
```
A=[1;2;3;1]
```

Out[6]:
```
4-element Array{Int64,1}:
1
2
3
1
```

In[7]:
```
B=[-1 2 1 4]
```

Out[7]:
```
1x4 Array{Int64,2}:
-1  2  1  4
```

In[8]:
```
(A*B)^10
```

Out[8]:
```
# Put your solution here

4x4 Array{Int64,2}:
-1000000000  2000000000  1000000000   4000000000
-2000000000  4000000000  2000000000   8000000000
-3000000000  6000000000  3000000000  12000000000
-1000000000  2000000000  1000000000   4000000000
```

In[9]:
```
B*A
```

Out[9]:
```
# Put your solution here

 1-element Array{Int64,1}:
10
```

and see if you can see and tell us what is going on.

Let us add one more computation

In[10]:
```
A*B
```

Out[10]:
```
4x4 Array{Int64,2}:
-1  2  1   4
-2  4  2   8
-3  6  3  12
-1  2  1   4
```

Notice that $(AB)^{10} = (AB) * 10^9$. This is because

$$(AB)^{10} = A(BA)^9 B = A(10)^9 B = AB(10)^9.$$

No computer would be needed.