NATIONAL UNIVERSITY OF SINGAPORE
Department of Mathematics

MA1506 Laboratory 3 (MATLAB)

Exercise 3

1. 716.25

2. 70. From the information available, $M = \begin{bmatrix} 0.7 & 0.2 & 0.3 \\ 0.2 & 0.5 & 0.3 \\ 0.1 & 0.3 & 0.4 \end{bmatrix}$. The 'brute force' way to compute the long run behaviour is to just calculate very large powers of M and compare the values of the entries.

   ```
   >> M^(200) - M^(100)
   >> M^(100)*[100; 100; 100]
   ```

   In the first command, we can see that the difference between the entries in M^{200} and M^{100} are less than 10^{-14}. So we are sufficiently confident that M^{100} is close enough to the long run value. Alternatively, we can do things properly by diagonalizing M and taking limits.

   ```
   >> [P D] =eig(M)
   >> D1= diag( [ 1 0 0 ])
   >> P*D1*inv(P)*[100; 100; 100]
   ```

   The **diag** command creates a diagonal matrix with the entries specified. It is clear that D1 is the limit of D^n as n gets large.

3. 104 million

4. Eigenvalues are -4, 3 and 3.

5. In this case we have complex eigenvalues $\pm i$ and note that P is also a matrix with complex entries.

6.
   ```
   >> C=[ 5 0 0; 1 5 0; 0 1 5]
   >> [P D] = eig(C)
   >> det(P)
   ```

   In this case, we have the eigenvalue 5 repeated 3 times, and the determinant of P is approximately zero. Actually, matrix A is not diagonalizable and the determinant of P should be exactly zero. This slight error is due to the algorithm used by the **eig** function.

7.
```
>> A=[5 6 2; 0 -1 -8; 1 0 -2]
>> poly(A)

ans =

    1.0000   -2.0000  -15.0000   36.0000

>> 1*A^3 -2*A^2 -15*A+36*eye(3)
>> B= [ -2 -1 ; 5 2]
>> poly(B)

ans =

    1.0000    0.0000    1.0000

>> 1*B^2 + eye(2)
>> C=[ 5 0 0; 1 5 0; 0 1 5]
>> poly(C)

ans =

     1   -15    75  -125

>> 1*C^3 -15*C^2 + 75*C -125*eye(3)
```

—The End—