Question 1

(a) Wilkinson Matrix

```
format long e
W = Wilkinson(5);
disp(W);
```

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

(b) Hamiltonian Matrix

```
H = Hamiltonian(3);
disp(H);
```

```
     9.441997267473083e-01    -7.043017284336089e-01     1.521013239005587e+00    -1.392409600065778e+00
    -2.120426688224212e+00    -1.018137216399071e+00    -3.843876388671156e-02     5.944630457374454e-01
    -6.446789155419369e-01    -1.820818684113852e-01     1.227447989009717e+00    -1.147380099446467e-01
     4.618574971904773e+00     1.437779497532469e+00     4.737103833615035e-01    -9.441997267473083e-01
     1.437779497532469e+00     1.118813577768040e-01    -1.112074968581539e+00     7.043017284336089e-01
     4.737103833615035e-01    -1.112074968581539e+00    -5.524357187095179e-01    -1.521013239005587e+00
```

Question 2

```
A = rand(8)
max_each_column = max(A)
max_each_row = max(A, [], 2)
max_overall = max(A, [], "all")
```

```
max_overall =
     9.897236392330852e-01
```

```
[row_indices, col_indices] = find(A > 0.25);
indices = [row_indices, col_indices]
```

Question 3

```
n_vals = [3, 5, 7];
for i=1:length(n_vals)
    x = n_vals(i);
    disp(['Magic Square for n = ', num2str(x)]);
    A = magic(x)
    col_sum = sum(A)
    row_sum = sum(A, 2)
    diagonal_sum = trace(A)
    antidiagonal_sum = trace(flipud(A))
end
```

```
Magic Square for n = 3
diagonal_sum =
    15
```

```
antidiagonal_sum =
    15
Magic Square for n = 5
diagonal_sum =
    65
antidiagonal_sum =
    65
Magic Square for n = 7
diagonal_sum =
   175
antidiagonal_sum =
   175
```

Question 4

```
disp('Question 4')
```

```
Question 4
```

```
x = 4;
A = magic(x)
sum(A)
sum(A')'
sum(diag(A))
```

```
ans =
    34
```

```
sum(diag(flipud(A)))
```

```
ans =
    34
```

```
rank(A)
```

```
ans =
     3
```

```
p = randperm(x);
q = randperm(x);
A = A(p, q);
sum(A)
sum(A')'
sum(diag(A))
```

```
ans =
    34
```

Run the experiment for some more time and check whether the diagonal sums change

```
sum(diag(flipud(A)))
```

```
ans =
    34
```

```
rank(A)
```

```
ans =
     3
```

observation ?

```
A = magic(4);
null(A)
null(A, 'r')
rref(A)
```

Question 5

```
A = rand(3);
A^(-1)
1./A
A.^(-1)
1./A
```

? observation ?

Question 6

```
p = [1 5 3 10 12]
syms x;
polynomial = poly2sym(p, x);
disp(polynomial);
```

$x^4 + 5\,x^3 + 3\,x^2 + 10\,x + 12$

```
pdash = (length(p)-1:-1:0).*p
polynomialdash = poly2sym(pdash(1:length(p)-1), x);
disp(polynomialdash);
```

$4\,x^3 + 15\,x^2 + 6\,x + 10$

It gives the coefficients of the differentiated polynomial.

Question 7

```
A1 = eye(16);
A2 = diag(ones(1, 15), 1);
A3 = diag(ones(1, 15), -1);
A = -2.*A1 + A2 + A3;
disp(A);
```

A(16,1) = 1
A(1,16) = 1.

```
    -2     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1    -2     1     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     1    -2     1     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     1    -2     1     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     1    -2     1     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     1    -2     1     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1    -2     1     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     1    -2     1     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     1    -2     1     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     1    -2     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     1    -2     1     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     1    -2     1     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     1    -2     1     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     1    -2     1     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     1    -2     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1    -2
```

```
toeplitz([-2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0])
temp = [1 2 3 4 5 6 7 8];
disp(triu(toeplitz(temp)))
```

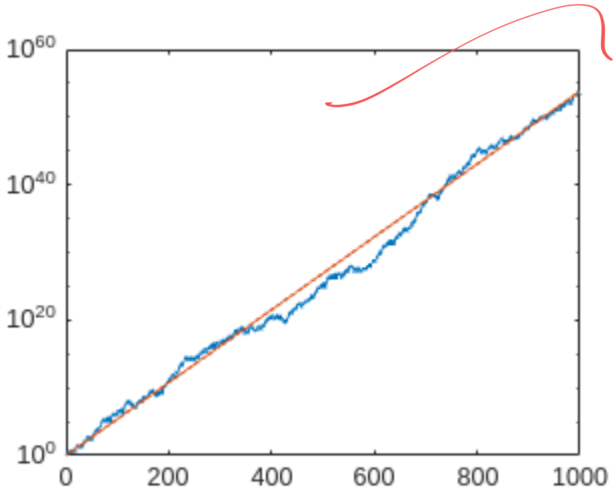
```
     1     2     3     4     5     6     7     8
     0     1     2     3     4     5     6     7
     0     0     1     2     3     4     5     6
     0     0     0     1     2     3     4     5
     0     0     0     0     1     2     3     4
     0     0     0     0     0     1     2     3
     0     0     0     0     0     0     1     2
     0     0     0     0     0     0     0     1
```

```
temp = 1./temp;
format rational
disp(toeplitz(temp))
```

```
       1            1/2          1/3          1/4          1/5          1/6          1/7
       1/2          1            1/2          1/3          1/4          1/5          1/6
       1/3          1/2          1            1/2          1/3          1/4          1/5
       1/4          1/3          1/2          1            1/2          1/3          1/4
       1/5          1/4          1/3          1/2          1            1/2          1/3
       1/6          1/5          1/4          1/3          1/2          1            1/2
       1/7          1/6          1/5          1/4          1/3          1/2          1
       1/8          1/7          1/6          1/5          1/4          1/3          1/2
```

Question 8

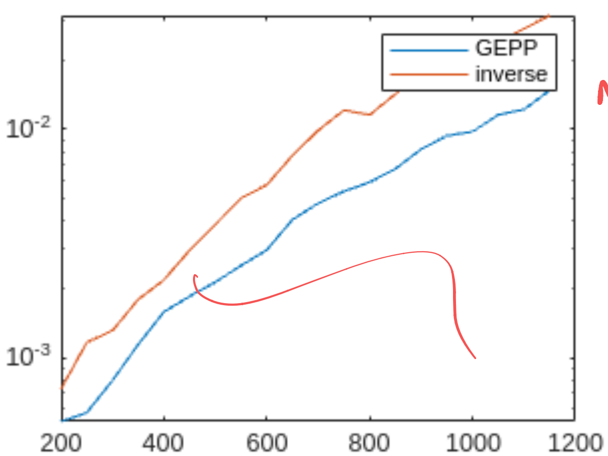
```
rand('state', 1000)
x = [1, 2];
for n=2:999, x(n+1) = x(n)+sign( rand-0.5)*x(n-1); end
semilogy (1:1000, abs(x))
c =1.13198824;
hold on
semilogy(1:1000, c.^ [1:1000])
hold off
```



This code tests the assertion $|x_n| = \mathcal{O}(c^n)$ by plotting both the curves and putting y axis in a logarithmic scale. From the graph we can verify that the assertion is true.

Question 9

```
format long e
rand('state', 1000)
time1 = zeros(1, 20);
time2 = zeros(1, 20);
matrix_sizes = zeros(1, 20);
for i = 200:50:1150
    matrix_sizes(i/50 - 3) = i;
    A = rand(i);
    b = rand(i, 1);
    tic
    x1 = A\b;
    time1(i/50 - 3) = toc;
    tic
    x2 = inv(A)*b;
    time2(i/50 - 3) = toc;
end
semilogy(matrix_sizes, time1, 'DisplayName', 'GEPP')
hold on
semilogy(matrix_sizes, time2, 'DisplayName', 'inverse')
hold off
legend('show')
```



Note that inv(A)*b takes more time.

As expected inverse method is taking more time than GEPP.

```
function W = Wilkinson(n)
    W = -1.*ones(5); n
    W = tril(W);
    W = W + 2.*eye(n);
    W(:, end) = 1;
```

```
end
function H = Hamiltonian(n)
    H11 = randn(n);
    H22 = -1.*transpose(H11);
    H12 = randn(n);
    H12 = H12 + transpose(H12);
    H21 = randn(n);
    H21 = H21 + transpose(H21);
    H = [H11 H12; H21 H22];
end
```