```
format long e
```

Question - 1

1 (a)

```
n=5;
W = Wilkinson(n)
```

1(b)

```
n=2;
H = Hamiltonian(n)
```

Question - 2

```
A = rand(8)
% maximum values in each column
max(A)
% maximum values in each row
max(A, [], 2)

% Max value of overall matrix
max(max(A))
```

```
ans =
     9.597439585160811e-01
```

```
% indices of elements larger than 0.25
[row, col] = find(A>0.25);
cat(2, row, col)
```

Question - 3

```
A = magic(4)
col_sum = sum(A)
row_sum = sum(A,2)
diag_sum = sum(diag(A))
```

```
diag_sum =
    34
```

```
A_flipped = flipud(A)
anti_diag_sum = sum(diag(A_flipped))
```

```
anti_diag_sum =
    34
```

We can observe that all the values are same here which confirms that the magic() works fine.

Question - 4

4 (a) and (b)

```
N = [3,4,5];

for n = N
    fprintf('For n = %d\n', n);
    A = magic(n)

    s1 = sum(A)
    s2 = sum(A')'
    s3 = sum(diag(A))
    s4 = sum(diag(flipud(A)))
    r = rank(A)

    fprintf('After Applying commands of part (b)\n');
    p = randperm(n); q = randperm(n);
    A = A(p,q)

    s1 = sum(A)
    s2 = sum(A')'
    s3 = sum(diag(A))
    s4 = sum(diag(flipud(A)))
    r = rank(A)
end
```

```
For n = 3
s3 =
    15
s4 =
    15
r =
     3
After Applying commands of part (b)
s3 =
     6
s4 =
    18
r =
     3
For n = 4
s3 =
    34
s4 =
    34
r =
     3
After Applying commands of part (b)
s3 =
    34
s4 =
    34
r =
     3
For n = 5
s3 =
    65
s4 =
```

```
    65
r =
     5
After Applying commands of part (b)
s3 =
    90
s4 =
    40
r =
     5
```

sum(A) prints sum of columns

sum(A')' prints sum of rows

sum(diag(A)) prints sum of diagonal elements of A

sum(diag(flipud(A))) prints sum of anti-diagonal elements of A

rank(A) prints rank of A

We observe that the commands in (b) part change the diagonal and anti-diagonal sum but the row, column sum and rank remains the same. (since it is random, sometimes the diagonal and anti-diagonal sum also turns out to be same)

4 (c)

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

Question - 5

```
n = 3;
A = rand(n)

A^(-1)

A.^(-1)

1./A
```

This shows that (a) is false and (b) is true.

Question - 6

Expected Observation: The given code prints a vector whose entries are the coefficients of the derivative of the given polynomial.

```
n = 4;
p = randn(1,n)

(length(p)-1:-1:0) .* p
```

Question - 7

7 (a)

```
arr1 = -2 * ones(1,16);
arr2 = ones(1,15);
arr3 = ones(1,1);

D = diag(arr1);
D = D + diag(arr2, 1) + diag(arr2, -1);
D = D + diag(arr3, 15) + diag(arr3, -15)
```

7 (b)

```
arr = zeros(1,16);
arr(1) = -2; arr(2) = 1; arr(16) = 1;
D = toeplitz(arr)
```

7 (c)

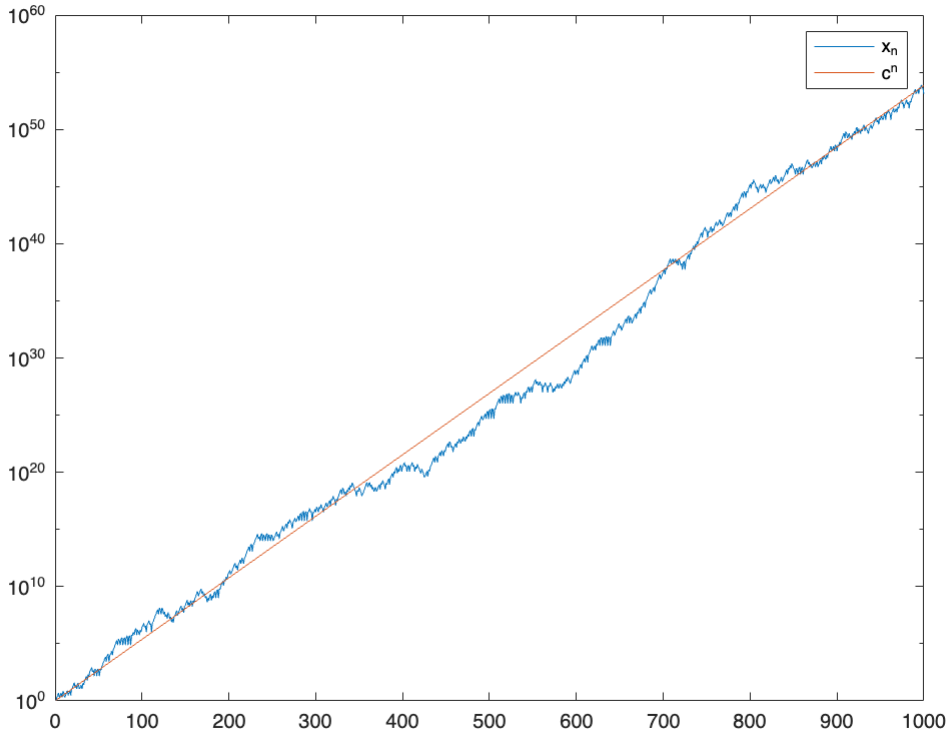
```
arr = [1,2,3,4,5,6,7,8];
D = triu(toeplitz(arr))
format rat
arr = 1 ./ arr;
D = toeplitz(arr)
```

```
D =
       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

figure;
semilogy (1:1000, abs(x))
c =1.13198824;
hold on
semilogy(1:1000, c.^[1:1000])
legend('x_n', 'c^n');
hold off;
```



Question - 9

```
sizes = 200:50:1150;
num_matrices = length(sizes);

time_backslash = zeros(1, num_matrices);
time_inv = zeros(1, num_matrices);

for i = 1:num_matrices
    n = sizes(i);

    A = rand(n);
    b = rand(n, 1);

    tic;
    x_backslash = A\b;
    time_backslash(i) = toc;

    tic;
    x_inv = inv(A) * b;
    time_inv(i) = toc;
end

% Plot the results using semilogy to plot log of the time taken
figure;
semilogy(sizes, time_backslash, '-o', 'DisplayName', 'A\\b');
```

```
hold on;
semilogy(sizes, time_inv, '-x', 'DisplayName', 'inv(A) * b');
hold off;

title('Comparison of Time Taken by A\\b and inv(A)*b');
xlabel('Matrix Size');
ylabel('Log of Time Taken (seconds)');

legend('Location', 'northwest');
grid on;
```

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

Functions

```
function W = Wilkinson(n)
    W = eye(n);
    W = W - tril(ones(n), -1);
    W(:, end) = ones(n,1);
end

function H = Hamiltonian(n)
    H_11 = randn(n);
    H_22 = -H_11';

    H_12 = randn(n);
    H_12 = H_12 + H_12';
```

```
    H_21 = randn(n);
    H_21 = H_21 + H_21';

    H = cat(1, cat(2, H_11, H_12), cat(2, H_21, H_22));
end
```