Zad 1
Utworzyć tablicę przechowującą sumy cząstkowe szeregu ∑(n=1,10) n.

```
% Zad 1
disp("# Zad 1")
A = [];
s = 0;
for i = 1:10
        s = s + i;
        A(i) = s;
end
wynik = A
```

Zad 2
Rozwiązać układ równań:
x1 +2x2 -x3 +3x4 = 7
-3x1 + x2 + x4 = 0
2x1 + x2 +x3 = 7
x1 - x2 +x3 + x4 = 1

```
% Zad 2
disp("# Zad 2")
R = [1  2 -1 3
        -3  1  0 1
         2  1  1 0
         1 -1  1 1];
W = [7; 0; 7; 1];
wynik = R\W
```

Zad 3
Przy pomocy wbudowanej funkcji quad obliczyć całkę ∫(1,2) sin(x)+x z dokładnością do 3 miejsca po przecinku.

```
% Zad 3
disp("# Zad 3")
xp = 1;
xk = 2;
tol = 3;
wynik = quad('fun1', xp, xk, tol)
%xp, xk przedziały całkowania
%tol dokładność
```

Do tego oddzielna funkcja:

```
function [y] = fun1(x)
        y = sin(x) + x;
endfunction
```

Zad 4
Napisać funkcję liczącą silnię liczby naturalnej n, korzystając z definicji silni i wywołać ją dla dowolnego n.

```
% Zad 4
disp("# Zad 4")
n = 6
wynik = silnia(n)
```

Do tego oddzielna funkcja:

```
function [k] = silnia(n)
  k = 1;
  for i = 1:n
    k = k * i;
  end
endfunction
```

main

```
% Zestaw 1

% Zad 1
disp("# Zad 1")
A = [];
s = 0;
for i = 1:10
        s = s + i;
        A(i) = s;
end
wynik = A

% Zad 2
disp("# Zad 2")
R = [1  2 -1 3
        -3  1  0 1
         2  1  1 0
         1 -1  1 1];

W = [7; 0; 7; 1];

wynik = R\W

% Zad 3
disp("# Zad 3")
xp = 1;
xk = 2;
tol =3;
wynik = quad('fun1', 1, 2, 2)
%xp, xk przedziały całkowania
%tol dokładność
% Zad 4
disp("# Zad 4")
n = 6
wynik = silnia(n)
```

do tego oddzielnie
funkcja fun1(x)
oraz silnia(n)

```
%Obliczanie funkcji
function k = fun1(x)
k = (x.^3 + x.^2 - 3.*x - 3) ;
endfunction

%porównywanie
function k = porownaj (a,b)
if(a<b)
   k=-1;
elseif(a>b)
   k=1;
else
   k=0;
end
endfunction

%macierz randomowa
function s = macierz (n)

A = round(10*rand(n))
suma = 0;
k = 1;
while (k <= n)
   suma = suma + A(k,k);
   k = k + 1;
endwhile
s = suma;
endfunction
```

Zad 1
Dla macierzy A = [1 2 3; 1 0 2] i dowolnie stworzonych przez siebie macierzy B i C wykonać:
a) mnożenie tablicowe macierzy A i B,
b) mnożenie macierzowe macierzy A i C.

```
% Zad 1
disp("# Zad 1")
A = [1 2 3; 1 0 2];
B = [2 4 5; 2 1 5];
C = [2 4; 5 3; 9 3];
disp("# a")
wynik = A .* B
disp("# b");
wynik = A * C
```

Zad 2
Dla szeregu ∑(n=1,∞) 1/n*sqrt(n) podać indeks liczby, której suma cząstkowa jest mniejsza od 0.001.

```
% Zad 2
disp("# Zad 2")
x = 1;
s = 0;
m = 0.001;
i = 0;
while(x >= m)
        i = i + 1;
        x = 1 / (i * sqrt(i));
        s = s + x;
end
indeks = i
wynik = s
```

Zad 3
Narysować wykres funkcji f(x)=sin(x)+cos(2x) w przedziale <0,8π> zielonymi gwiazdkami.

```
% Zad 3
disp("# Zad 3")
x = 0:0.1:8*pi;
y = sin(x) + cos(2*x);
plot(x,y,'g*');
title("sin(x)+cos(2x)");
xlabel('x');
ylabel('y');
```

Zad 4
Napisać funkcję obliczającą ciąg Fibonacciego i wywołać ją dla n = 20.

```
% Zad 4
disp("# Zad 4")
n = 20
wynik = fib(n)
```

Do tego oddzielna funkcja:

```
function [f] = fib(n)
  if(n <= 0)
        f = 0;
    elseif(n == 1)
        f = 1;
    else
        f = fib(n-1) + fib(n-2);
    end
endfunction
```

main

```
% Zestaw 2

% Zad 1
disp("# Zad 1")
A = [1 2 3; 1 0 2];
B = [2 4 5; 2 1 5];
C = [2 4; 5 3; 9 3];
disp("# a")
wynik = A .* B
disp("# b");
wynik = A * C

% Zad 2
disp("# Zad 2")
x = 1;
s = 0;
m = 0.001;
i = 0;
while(x >= m)
        i = i + 1;
        x = 1 / (i * sqrt(i));
        s = s + x;
end
indeks = i
wynik = s

% Zad 3
disp("# Zad 3")
x = 0:0.1:8*pi;
y = sin(x) + cos(2*x);
plot(x,y,'g*');
title("sin(x)+cos(2x)");
xlabel('x');
ylabel('y');

% Zad 4
disp("# Zad 4")
n = 20
wynik = fib(n)
```

do tego oddzielnie funkcja fib(n)

Zad 1
Wykonać mnożenie tablicowe podanych macierzy:
A = [1 2 3; 1 0 1],
B = [2 1 2; 0 2 3].

```
% Zad 1
disp("# Zad 1")
A = [1 2 3; 1 0 1];
B = [2 1 2; 0 2 3];
wynik = A .* B
```

Zad 2
Dla szeregu ∑(n=1,∞) 1/n podać indeks liczby, kiedy suma cząstkowa przekroczy liczbę 5.

```
% Zad 2
disp("# Zad 2")
x = 1;
s = 0;
m = 5;
i = 0;
while(s < m)
        i = i + 1;
        x = 1 / i;
        s = s + x;
end
indeks = i
wynik = s
```

Zad 3
Narysować niebieską linią wykres wielomianu interpolacyjnego trzeciego stopnia przechodzącego przez punkty (0,0),(1,1),(2,0) zaznaczone czerwonymi okręgami.

```
% Zad 3
disp("# Zad 3")
x0 = [0 1 2];
y0 = [0 1 0];
w = 3;
p = polyfit(x0,y0,w)
x = -3:0.1:3;
y = polyval(p,x);
plot(x0,y0,'ro',x,y,'b');
title("Interpolacja 3 stopnia");
xlabel('x');
ylabel('y');
```

Zad 4
Napisać funkcję szukającą litery w podanym ciągu znaków i wywołać ją dla poniższych argumentów:
L = "Teraz masz zdac",
z = 'c'.

```
% Zad 4
disp("# Zad 4")
L = "Teraz masz zdac";
z = 'c';
wynik = szukaj(L,z)
```

Do tego oddzielna funkcja:

```
function k = szukaj(L,z)
        k = 0;
        s = size(L);
        s = s(2);
        for i = 1:s
                if L(i) == z
                        k = i;
                        break;
                end
        end
endfunction
```

main

```
% Zestaw 3

% Zad 1
disp("# Zad 1")
A = [1 2 3; 1 0 1];
B = [2 1 2; 0 2 3];
wynik = A .* B

% Zad 2
disp("# Zad 2")
x = 1;
s = 0;
m = 5;
i = 0;
while(s < m)
        i = i + 1;
        x = 1 / i;
        s = s + x;
end
indeks = i
wynik = s

% Zad 3
disp("# Zad 3")
x0 = [0 1 2];
y0 = [0 1 0];
w = 3;
p = polyfit(x0,y0,w)
x = -3:0.1:3;
y = polyval(p,x);
%plot(x0,y0,'ro',x,y,'b');
%title("Interpolacja 3 stopnia");
%xlabel('x');
%ylabel('y');

% Zad 4
disp("# Zad 4")
L = "Teraz masz zdac";
z = 'c';
wynik = szukaj(L,z)
```

do tego oddzielnie funkcja szukaj(L,z)

Zad 1
Zdefiniować poniższe macierze:
A = [1 1 1; 1 1 1],
B = [2; 2],
C = [3 3 3 3],
Złożyć macierz D z podanych powyżej macierzy w taki sposób, aby wyglądała następująco:

```
    | 1 1 1 2 |
D = | 1 1 1 2 |
    | 3 3 3 3 |
```

```
% Zad 1
disp("# Zad 1")
A = [1 1 1; 1 1 1];
B = [2; 2];
C = [3 3 3 3];
D = [A, B; C];
wynik = D
```

Zad 2
Usunąć drugi wiersz macierzy D.

```
% Zad 2
disp("# Zad 2")
D(2,:) = [];
wynik = D
```

Zad 3
Funkcja przyjmuje poniższe wartości:
y = {3*x^2 dla x < 0
{40*sqrt(x) dla x >= 0
Narysować czerwoną linią wykres tej funkcji w przedziale <-9,0> z krokiem 0.5 oraz <1,15> z krokiem 1.

```
% Zad 3
disp("# Zad 3")
x = [[-9:0.5:0],[1:15]];
y = fun1(x);
figure(1);
plot(x,y,'r');
title("Fun1");
xlabel('x');
ylabel('y');
Do tego oddzielna funkcja:
function [y] = fun1(x)
        s = size(x);
        s = s(2);
        for i = 1:s
                if x(i) < 0
                        y(i) = 3 * x(i)^2;
                else
                        y(i) = 40 * sqrt(x(i));
                end
        end
endfunction
```

Zad 4
Napisać funkcję, która rysuje na wykresie czarnymi gwiazdkami choinkę, w zależności od ilości wierszy.
Przykład dla n = 5:

```
 * * * * *
  * * * *
   * * *
    * *
     *
```

```
% Zad 4
disp("# Zad 4")
n = 5
figure(2);
choinka(n)
Do tego oddzielna funkcja:
function choinka(n)
        if n > 0
                x = [];
                y = [];
                k = 0;
                for i = 1:n
                        for j = 1:i
                                k = k + 1;
                                x(k) = -j;
                                y(k) = i;
                        end
                end
                plot(x,y,'k*');
                axis([-(n+1) 0 0 n+1]);
                title("Choinka");
        end
endfunction


main
% Zestaw 4

% Zad 1
disp("# Zad 1")
A = [1 1 1; 1 1 1];
B = [2; 2];
C = [3 3 3 3];
D = [A, B; C];
wynik = D

% Zad 2
disp("# Zad 2")
D(2,:) = [];
wynik = D

% Zad 3
disp("# Zad 3")
x = [[-9:0.5:0],[1:15]];
y = fun1(x);
figure(1);
plot(x,y,'r');
title("Fun1");
xlabel('x');
ylabel('y');
do tego oddzielnie funkcja fun1(x)

% Zad 4
disp("# Zad 4")
n = 5
figure(2);
choinka(n)
do tego oddzielnie funkcja choinka(n)
```