1. Implementati functii ce genereaza formule de cuadratura gaussiene pentru ponderile clasice date in tabela 1.

Solutie: A se vedea functiile de la finalul documentului.

```
f=@(x) exp(x)
```

```
f = function_handle with value:
    @(x)exp(x)
```

```
GaussQuadIntPrecision(@(x) f(x), @Gauss_Legendre, 50, 1e-4)
```

```
Precision reached after 4 epochs
ans = 2.3504
```

```
GaussQuadIntPrecision(@(x) f(x).*sqrt(1-x.^2), @Gauss_Cebisev1, 50, 1e-4)
```

```
Precision reached after 30 epochs
ans = 2.3518
```

```
GaussQuadIntPrecision(@(x) f(x)./sqrt(1-x.^2) , @Gauss_Cebisev2, 50, 1e-4)
```

```
Precision reached after 37 epochs
ans = 2.3486
```

```
real = integral(f, -1, 1)
```

```
real = 2.3504
```

```
global Laguerre_a
Laguerre_a = 2;
f=@(x) x.^2 .* exp(-x) .* sin(x)
```

```
f = function_handle with value:
    @(x)x.^2.*exp(-x).*sin(x)
```

```
GaussQuadIntPrecision(@(x) sin(x), @Gauss_Laguerre_wo_param, 50, 1e-4)
```

```
Precision reached after 11 epochs
ans = 0.5000
```

```
real = integral(f,0, 10000)
```

```
real = 0.5000
```

```

```

2. Aproximati

$$\int_{-1}^{1} \sin(x^2)dx$$

printr-o formula Gauss-Legendre si comparati rezultatul cu cel returnat de quad sau quadl

```
int_gauss_leg = GaussQuadIntPrecision(@(x) sin(x.^2), @Gauss_Legendre, 50, 1e-16)
```

```
Precision reached after 17 epochs
int_gauss_leg = 0.6205
```

```
int_quad = quad(@(x) sin(x.^2), -1, 1)
```

```
int_quad = 0.6205
```

```
err = abs(int_gauss_leg-int_quad)
```

```
err = 4.7619e-08
```

3. Calculati

$$\int_{\mathbb{R}} e^{-x^2} \sin(x)dx, \int_{\mathbb{R}} e^{-x^2} \cos(x)dx,$$

utilizând o cuadratură Gauss-Hermite.

```
int_s = GaussQuadIntPrecision(@(x) sin(x), @Gauss_Hermite, 50, 1e-10)
```

```
Precision reached after 2 epochs
int_s = 0
```

```
int_c = GaussQuadIntPrecision(@(x) cos(x), @Gauss_Hermite, 50, 1e-10)
```

```
Precision reached after 9 epochs
int_c = 1.3804
```

```

real_int_s = quad(@(x) exp(-x.^2).*sin(x), -10000, +10000) % -inf..inf
```

```
real_int_s = 0
```

```
real_int_c = quad(@(x) exp(-x.^2).*cos(x), -10000, +10000)
```

```
real_int_c = 1.3804
```

Helpers:

```
function I=GaussQuadIntPrecision(f, nodes_generator, iters, tol)
    I = GaussQuadInt(f, nodes_generator, 1);
    I0 = I;
    for i=2:iters
        I = GaussQuadInt(f, nodes_generator, i);
        if(abs(I-I0)<tol)
            fprintf("Precision reached after %i epochs", i);
            return
        end
        I0=I;
    end
    fprintf("Required precision was not reached. Error is %f", abs(I-I0));
```

```
end
```

```
function I=GaussQuadInt(f, nodes_generator, n)
    [g_nodes, g_coeff] = nodes_generator(n);
    I = vquad(g_nodes, g_coeff, f);
end
```

```
function I = vquad(g_nodes, g_coeff, f)
    I = g_coeff * f(g_nodes);
end
```

```
% Gauss-Cebîșev de speța I
function [g_nodes, g_coeff] = Gauss_Cebisev1(n)
    g_coeff = pi / n * ones(1, n);
    g_nodes = cos(pi * ([1 : n]' - 0.5) / n);
end
```

```
% Gauss-Cebîșev de speța a II-a.
function [g_nodes, g_coeff] = Gauss_Cebisev2(n)
    beta = [pi / 2, 1 / 4 * ones(1, n - 1)];
    alpha = zeros(n, 1);
    [g_nodes, g_coeff] = GaussQuad(alpha, beta);
end
```

```
function [g_nodes, g_coeff] = Gauss_Hermite(n)
    beta = [sqrt(pi), [1 : n - 1] / 2];
    alpha = zeros(n, 1);
    [g_nodes, g_coeff] = GaussQuad(alpha, beta);
end
```

```
function [g_nodes, g_coeff] = Gauss_Legendre(n)
    beta = [2, (4 - ([1 : n - 1]).^(-2)).^(-1)];
    alpha = zeros(n, 1);
    [g_nodes, g_coeff] = GaussQuad(alpha, beta);
end
```

```
function [g_nodes, g_coeff] = Gauss_Jacobi(n, a, b)

    k = 0 : n - 1;
```

```
    k2 = 2 : n - 1;

    bet1 = 4 * (1 + a) * (1 + b) / ((2 + a + b)^2) / (3 + a + b);
    beta = [2^(a + b + 1) * beta(a + 1, b + 1), bet1, 4 * k2 .* (k2 + a + b) .* (k2 + a) .* (k2

    if (a == b)
        alpha = zeros(1, n);
    else
        alpha = (b^2 - a^2) ./ (2 * k + a + b) ./ (2 * k + a + b + 2);
    end

    [g_nodes, g_coeff] = GaussQuad(alpha, beta);
end
```

```
function [g_nodes, g_coeff] = Gauss_Laguerre_wo_param(n)
    global Laguerre_a
    [g_nodes, g_coeff] = Gauss_Laguerre(n, Laguerre_a);
end
```

```
function [g_nodes, g_coeff] = Gauss_Laguerre(n, a)
    k = 1 : n - 1;
    alpha = [a + 1, 2 * k + a + 1];
    beta = [gamma(1 + a), k .* (k + a)];
    [g_nodes, g_coeff] = GaussQuad(alpha, beta);
end
```

```
function [q_nodes, q_coeff] = GaussQuad(alpha, beta)

    n = length(alpha);
    rb = sqrt(beta(2 : n));
    J = diag(alpha) + diag(rb, -1) + diag(rb, 1);
    [v, d] = eig(J);
    q_nodes = diag(d);
    q_coeff = beta(1) * v(1, :).^2;

end
```