```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double pi = 3.1415926535;
double e = 2.7182818284;

// y"+y=1, y(0) = 0, y(pi/2) = 0
double f1(double x) {return 1;}
double p1(double x) {return 0;}
double q1(double x) {return 1;}
double koef_data_1[6] = {1, 0, 0, 1, 0, 0}; // sigma1 gamma1 delta1 / sigma 2 gamma2 delta2
double ab_data_1[2] = {0, 3.1415926535/2};
double y1_exac(double x) {return 1 - sin(x) - cos(x);} // y = 1 -sinx - cosx

// y"-y=2x, y(0) = 0, y(1) = -1
double f2(double x) {return 2.*x;}
double p2(double x) {return 0;}
double q2(double x) {return -1;}
double koef_data_2[6] = {1, 0, 0, 1, 0, -1};
double ab_data_2[2] = {0, 1};
double y2_exac(double x) {return sinh(x)/sinh(1) - 2*x;} // y = sh(x)/sh(1) - 2x

// y"-y'=0, y(0) = -1, y'(1) - y(1) = 2
double f3(double x) {return 0;}
double p3(double x) {return -1;}
double q3(double x) {return 0;}
double koef_data_3[6] = {1, 0, -1, -1, 1, 2};
double ab_data_3[2] = {0, 1};
double y3_exac(double x) {return pow(2.7182818284, x) - 2;} // y = e^x - 2

double ab_data[6] = {0, 3.1415926535/2, 0, 1, 0, 1}; // common data
double koef_data[18] = {1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, -1, 1, 0, -1, -1, 1, 2}; // common koef

double ((*function_f[3]))() = {f1, f2, f3};
double (*function_p[3])() = {p1, p2, p3};
double (*function_q[3])() = {q1, q2, q3};
double (*function_exac[3])() = {y1_exac, y2_exac, y3_exac};

void sweep_method(double *y, double *alpha, double *beta, double s2, double g2, double d2,
                  double h, int n);

void alpha_beta_search(double *alpha, double *beta, double a, double h, double s1,
                       double g1, double d1, double (*p)(double), double (*q)(double),
                       double (*f)(double), int n);

int main(int argc, char **argv) // на вход номер ф-ции и кол-во иттераций n
{
    int fnum, n;
    sscanf(argv[1], "%d", &fnum); fnum--;
    sscanf(argv[2], "%d", &n);

    double ((*f))()  = function_f[fnum];
    double ((*p))()  = function_p[fnum];
    double ((*q))()  = function_q[fnum];

    double a = ab_data[fnum*2];
    double b = ab_data[fnum*2+1];
    double h = (b-a)/n;

    //sigma gamma delta
    double s1 = koef_data[fnum*6 + 0];
    double g1 = koef_data[fnum*6 + 1];
    double d1 = koef_data[fnum*6 + 2];
    double s2 = koef_data[fnum*6 + 3];
    double g2 = koef_data[fnum*6 + 4];
    double d2 = koef_data[fnum*6 + 5];

    // y - solution; alpha, beta - koef
    double *y = calloc(n+1, sizeof(double));
    double *alpha = calloc(n+1, sizeof(double));
    double *beta = calloc(n+1, sizeof(double));

    alpha_beta_search(alpha, beta, a, h, s1, g1, d1, p, q, f, n);
    sweep_method(y, alpha, beta, s2, g2, d2, h, n);

    char name[128];
    sprintf(name, "test_%d.txt", fnum+1);
    FILE *out = fopen(name, "w");

    double ((*exac))()  = function_exac[fnum];
    for(int i = 0; i < n + 1; i++) {
        double x = a + h * i;
        fprintf(out, "%9.3lf %9.3lf %9.3lf\n", x, y[i], exac(x));
    }
    return 0;
}

void alpha_beta_search(double *alpha, double *beta, double a, double h, double s1,
                       double g1, double d1, double (*p)(double), double (*q)(double),
                       double (*f)(double), int n)
{
    alpha[1] = -1. * (g1)  / (s1 - g1); //&
    beta[1] = (d1 * h) / (s1 - g1);
    for(int i = 1; i < n; i++) {
        double x = a + i * h;
        double P = p(x);
        double Q = q(x);
        double F = f(x);

        alpha[i + 1] = (1 / (h*h) + P / (2*h)) /
                ((2./(h*h) - Q) - (1./(h*h) - P/(2.*h)) * alpha[i]);
        beta[i + 1] = (beta[i]*(1/(h*h) - P/(2*h)) - F) /
                ((2./(h*h) - Q) - (1./(h*h) - P/(2.*h)) * alpha[i]);
    }
    return;
}
void sweep_method(double *y, double *alpha, double *beta, double s2, double g2, double d2,
                  double h, int n)
{
    y[n] = (g2 * beta[n] + d2 * h) / (g2 * (1 - alpha[n]) + s2 * h);
    for (int i = n - 1; i >= 0; i--) {
        y[i] = y[i + 1] * alpha[i + 1] + beta[i + 1];
    }
    return;
}
```