

```

/*
 * Barakat Soror Al-Rasheedi (321100797)
 *
 * Artificial neural network for XOR.
 */
#include <stdio.h>
#include <math.h>

#define LOG_FILE    0
#define ALPHA      0.1
#define ACCEPTED    10e-4
#define sigmoid(x)  (1.0 / (1.0 + exp(-(x))))

int main(void)
{
    /* Weights */
    double w13 = 0.5;
    double w14 = 0.9;
    double w23 = 0.4;
    double w24 = 1.0;
    double w35 = -1.2;
    double w45 = 1.1;
    /* Thresholds. */
    double theta3 = 0.8;
    double theta4 = -0.1;
    double theta5 = 0.3;
    /* Final neuron output. */
    double y5;
    /* Error. */
    double e;
    /* Sum of squared errors. */
    double sse;
    /* For logging. */
    unsigned epoch = 1;
    unsigned x1;
    unsigned x2;
    unsigned yd;

    #if LOG_FILE
        /* Log file for epoch and sum of squared errors. */
        FILE *log;
        log = fopen("log.dat", "w");
        if(log == NULL) {
            perror("Error");
            return -1;
        }
    #endif

    for(; ; epoch++) {
        unsigned i;
        sse = 0.0;

        for(i = 0 ; i < 4 ; i++) {
            /* Outputs of internal neurons. */
            double y3;

```

```

double y4;
/* Error gradient. */
double eg5;
double eg3;
double eg4;

    if(i == 0) { x1 = 1; x2 = 1; }
else if(i == 1) { x1 = 0; x2 = 1; }
else if(i == 2) { x1 = 1; x2 = 0; }
else if(i == 3) { x1 = 0; x2 = 0; }

yd = x1 ^ x2;

/* Compute the outputs. */
y3 = sigmoid(x1 * w13 + x2 * w23 - theta3);
y4 = sigmoid(x1 * w14 + x2 * w24 - theta4);
y5 = sigmoid(y3 * w35 + y4 * w45 - theta5);
/* Compute the error. */
e = yd - y5;
/* Compute the error gradient per output. */
eg5 = y5 * (1.0 - y5) * e;
eg3 = y3 * (1.0 - y3) * eg5 * w35;
eg4 = y4 * (1.0 - y4) * eg5 * w45;
/* Add to each weight its delta. */
w35 += ALPHA * y3 * eg5;
w45 += ALPHA * y4 * eg5;
w13 += ALPHA * x1 * eg3;
w23 += ALPHA * x2 * eg3;
w14 += ALPHA * x1 * eg4;
w24 += ALPHA * x2 * eg4;
/* Add to each threshold its delta. */
theta3 += ALPHA * (-1) * eg3;
theta4 += ALPHA * (-1) * eg4;
theta5 += ALPHA * (-1) * eg5;
/* Add the the square of the error. */
sse += e * e;
#ifdef LOG_FILE
    fprintf(log, "%.4f ", sse);
#endif
}
/* Stop if we reached an accepted SSE. */
if(sse <= ACCEPTED) break;
}
/* Log the results of the last epoch. */
printf("Epochs: %u\nx1 = %u, x2 = %u, yd = %u, y5 = %0.4f, e = %0.4f\n"
"w13 = %0.4f, w14 = %0.4f, w23 = %0.4f, w24 = %0.4f, w35 = %0.4f, w45 = %0.4f\n"
"theta3 = %0.4f, theta4 = %0.4f, theta5 = %0.4f\n"
"Sum of squared errors = %0.4f\n\n",
epoch, x1, x2, yd, y5, e, w13, w14, w23, w24, w35, w45,
theta3, theta4, theta5, sse);
#ifdef LOG_FILE
    fclose(log);
#endif
return 0;
}

```