```c
#include <stdio.h>
#include <stdlib.h>     // For the rand function
#include <time.h>       // For the time function

double UniformDist (double a, double b){ return a + (b - a) * (((double) rand ()) /         ↲
((double) RAND MAX)); }
double tobounded (double x, double bound){ return ((x > bound) ? bound : ((x < -bound) ?    ↲
-bound : x)); }

#define ZMAX(x) (((x) > 0) ? (x) : 0.0)
#define GAMMA(x) ((x < 1) ? x : x*x)    // NOTE: This is already q i^{gamma(q i)}
#define THETA(x) ((x < 0.001) ? 10 : ((x <= 0.1) ? 20 : ((x <= 1) ? 100 : 300)))

double f (double *x) {  return -(
       x[0] * (10.5 + 0.5 * x[0]) +
       x[1] * (7.5 + 0.5 * x[1]) +
       x[2] * (3.5 + 0.5 * x[2]) +
       x[3] * (2.5 + 0.5 * x[3]) +
       x[4] * (1.5 + 0.5 * x[4]) + 10 * x[5]);
} // y = x[5]

void qilist (double *x, double *qi) { // Convenient for penalty method
  qi[0] = ZMAX (6 * x[0] + 3 * (x[1] + x[2]) + 2 * x[3] + x[4] - 6.5);
  qi[1] = ZMAX (10 * (x[0] + x[2]) + x[5] - 20.0);  // This gives y <= 20.0
  qi[2] = (x[0] > 1) ? x[0] - 1 : 0.0;  // qi[2] = x[0] - 1
  qi[3] = (x[1] > 1) ? x[1] - 1 : 0.0;  // qi[3] = x[1] - 1
  qi[4] = (x[2] > 1) ? x[2] - 1 : 0.0;  // qi[4] = x[2] - 1
  qi[5] = (x[3] > 1) ? x[3] - 1 : 0.0;  // qi[5] = x[3] - 1
  qi[6] = (x[4] > 1) ? x[4] - 1 : 0.0;  // qi[6] = x[4] - 1
  qi[7] = (x[0] < 0) ? -x[0] : 0.0;     // qi[7]  = -x[0]
  qi[8] = (x[1] < 0) ? -x[1] : 0.0;     // qi[8]  = -x[1]
  qi[9] = (x[2] < 0) ? -x[2] : 0.0;     // qi[9]  = -x[2]
  qi[10] = (x[3] < 0) ? -x[3] : 0.0;    // qi[10] = -x[3]
  qi[11] = (x[4] < 0) ? -x[4] : 0.0;    // qi[11] = -x[4]
  qi[12] = (x[5] < 0) ? -x[5] : 0.0;    // qi[12] = -y = -x[5]
}
double Fit (double *x, double ck){
  register unsigned char i;
  double q[13], H = 0.0;

  qilist (x, q);
  for (i = 0; i < 13; i++) if (q[i] > 0.0) H += THETA (q[i]) * GAMMA (q[i]);
  return f (x) + ck * H;
}

/*  Solution is: x[0] = x[2] = 0.0; x[1] = x[3] = x[4] = 1.0; x[5] = 20;
 *  with Fitness = -213 */

#define S 100
#define C1 2
#define C2 2
#define VMAX 4.0
#define OmegaIni 1.2
#define OmegaFi 0.1
#define MAX PENALTY NITER 1000
#define MAX NITER FO 5
#define ckini 1.0

void main () {
  double x[S][7], p[S][7], V[S][6], gx[6];
  double Domega = (OmegaIni - OmegaFi)/(MAX PENALTY NITER-1);
  unsigned niter = 0U;
  register unsigned i, j, r;

  srand (time (0));     // Randomization

// Initialisation
  for (i = 0; i < S; i++) {
    for (j = 0; j < 5; j++) {
      p[i][j] = x[i][j] = UniformDist (0.0, 1.0);
      V[i][j] = UniformDist (-1.0, 1.0);
    }
    p[i][5] = x[i][5] = UniformDist (0.0, 20.0);
    V[i][5] = UniformDist (-VMAX, VMAX);
    p[i][6] = x[i][6] = Fit (&(x[i][0]), ckini);
  }

// Swarm best
  i= 0U; double fitmin = x[i][6]; for (j = 1; j < S; j++) if (x[j][6] < fitmin) { i = j; fitmin = x[i][6]; }
  for (j = 0; j < 6; j++) gx[j] = x[i][j];

// Main loop
  while (niter < MAX PENALTY NITER){
    double ck = ckini + niter*niter, omega = OmegaIni - niter*Domega;
    for(r=0; r < MAX NITER FO; r++){
      for (i = 0; i < S; i++) {
        double r1 =  UniformDist (0.0, C1), r2 = UniformDist (0.0, C2);
        for (j = 0; j < 6; j++) {
          V[i][j] = tobounded(omega*V[i][j] + r1*(p[i][j] - x[i][j]) + r2*(gx[j] - x[i][j]), VMAX);
          x[i][j] += V[i][j];
        }
        x[i][6] = Fit (&(x[i][0]), ck); // Compute fitnes of x i
        if(x[i][6] < p[i][6]) { for (j = 0; j < 7; j++) p[i][j] = x[i][j] ; } // Update iteration best of particle i
      }

// Swarm best
      i= 0U; fitmin = x[i][6]; for (j = 1; j < S; j++) if (x[j][6] < fitmin) { i = j; fitmin = x[i][6]; }
      for (j = 0; j < 6; j++) gx[j] = x[i][j];
    }
    niter ++;
  }

  printf("Solution found: x 1 = %g; x 2 = %g; x 3 = %g; x 4 = %g; x 5 = %g; y = %g; Fitness = %g\n", gx[0], gx[1], gx[2], gx[3], gx[4], gx[5], f(gx));
}
```