```c
/*
 * Other online mathematics problems (c)
 * August 27, 2024 by James Pate Williams, Jr.
 * Find the root of f(t) = cos(2t) + sin(3t)
 * t    =  0.628318530718
 * f(t) =  1.11022302e-16
 * Also find the roots of the equations
 * f(x) = sqrt(1 + sqrt(1 + x)) - x^1/3
 * x    =  8.000000000000
 * f(x) =  0.00000000e+00
 * Solution of f(x) = 9^x + 12^x - 16^x
 * x    =  -16.387968065352
 * f(x) =    2.32137533e-16
 * Solution of f(x) = 8^x-2^x - 2(6^x-3^x)
 * x    =  1.000000000000
 * f(x) =  0.00000000e+00
 * Solution of f(a,x)=sin(sqrt(ax-x^2))=0
 * Subject to the constraint x+y=100
 * Where x and y are the two roots of
 * g(a,x)=ax-x^2-n*n*pi*pi=0\n") and n=15
 * Solve for a real root of the equation
 * f(x)=log6l(5+x)+log6l(x)=0
 */

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef long double real;

real f1(real t) {
    return cosl(2 * t) + cosl(3 * t);
}

real g(real t) {
    return -2 * sinl(2 * t) - 3 * sinl(3 * t);
}

real f2(real x) {
    return sqrtl(1 + sqrtl(1 + x)) - powl(x, 1.0 / 3.0);
}

real f3(real x) {
    return powl(9, x) + powl(12, x) - powl(16, x);
}

real f4(real x) {
    return powl(8, x) - powl(2, x) -
        2 * (powl(6, x) - powl(3, x));
}

```

```
real CentralDifference(
    real x, real h, real(*fx)(real)) {
    return (fx(x + h) - fx(x - h)) / (2.0 * h);
}

real NewtonRaphson1(
    real x0, real ftolerance, real xtolerence,
    real(*fx)(real), real(*deriv)(real)) {
    real x1 = 0.0;
    while (1) {
        real f = fx(x0);
        real dfdx = deriv(x0);
        x1 = x0 - f / dfdx;
        if (fabsl(x1 - x0) < xtolerence ||
            fabsl(f) < ftolerance)
            break;
        x0 = x1;
    }
    return x1;
}

real NewtonRaphson2(
    real x0, real ftolerance, real xtolerence,
    real(*fx)(real),
    real(*deriv)(real, real, real(*fx)(real))) {
    real h = 2.0 / 10000.0, x1 = 0.0;
    while (1) {
        real f = fx(x0);
        real dfdx = deriv(x0, h, fx);
        x1 = x0 - f / dfdx;
        if (fabsl(x1 - x0) < xtolerence ||
            fabsl(f) < ftolerance)
            break;
        x0 = x1;
    }
    return x1;
}

void QuadraticEquation(
    real a, real b, real c,
    real* x, real* y) {
    real d = b * b - 4 * a * c;
    if (d >= 0) {
        *x = (-b + sqrtl(d)) / (2.0 * a);
        *y = (-b - sqrtl(d)) / (2.0 * a);
    }
    else
        *x = *y = 0.0;
}

real f5(real a, real x) {
    return sinl(sqrtl(a * x - x * x));
```

```
105 }
106
107 real f6(int n, real a, real x) {
108     real pi = 4.0 * atanl(1.0);
109     return a * x - x * x - (real)n * n * pi * pi;
110 }
111
112 real f7(int n, real a, real x) {
113     return sqrt(f6(n, a, x));
114 }
115
116 void HillClimber(
117     int n, int population, int generation,
118     int seed, real* a, real* f, real* x,
119     real* y) {
120     int g = 0, i = 0, j = 0;
121     int* index = (int*)calloc(
122         population, sizeof(int));
123     real bestFitness = 1.0e16;
124     real childa = 0, childx = 0, childy = 0;
125     real pi = 4.0 * atanl(1.0);
126     real* fitness = (real*)calloc(
127         population, sizeof(real));
128     real** gene = (real**)calloc(
129         population, sizeof(real*));
130     if (fitness == NULL)
131         exit(-1);
132     if (gene == NULL)
133         exit(-2);
134     for (i = 0; i < population; i++) {
135         gene[i] = (real*)calloc(3, sizeof(real));
136         if (gene[i] == NULL)
137             exit(-3);
138     }
139     for (i = 0; i < population; i++) {
140         childa = gene[i][0] = 112.0 * (real)rand() / RAND_MAX;
141         QuadraticEquation(
142             -1.0, childa, -(real)n * n * pi * pi,
143             &childx, &childy);
144         fitness[i] = f6(n, childa, childx);
145         gene[i][1] = childx;
146         gene[i][2] = childy;
147         *a = childa;
148         *x = childx;
149         *y = childy;
150     }
151     for (g = 1; g <= generation; g++) {
152         int count = 0;
153         int parent1 = rand() % population;
154         int parent2 = rand() % population;
155         int parentc = parent1 < parent2 ?
156             parent1 : parent2;
```

```
157         real maximum = fitness[0];
158         childa = gene[parentc][0] + 0.0001;
159         QuadraticEquation(
160             -1.0, childa, -(real)n * n * pi * pi,
161             &childx, &childy);
162         real childf = f6(n, childa, childx);
163         for (i = 1; i < population; i++)
164             if (fitness[i] > maximum)
165                 maximum = fitness[i];
166         for (i = 0; i < population; i++)
167             if (fitness[i] == maximum)
168                 index[count++] = i;
169         if (childf < fitness[index[rand() % count]]) {
170             gene[parentc][0] = childa;
171             gene[parentc][1] = childx;
172             gene[parentc][2] = childy;
173             fitness[parentc] = childf;
174         }
175         if (childf < bestFitness &&
176             childf >= 0.0) {
177             *a = childa;
178             *x = childx;
179             *y = childy;
180             *f = childf;
181             bestFitness = childf;
182         }
183     }
184 }
185 
186 real log6l(real x) {
187     return log10l(x) / log10l(6.0);
188 }
189 
190 real f8(real x) {
191     return log6l(5 + x) + log6l(x);
192 }
193 
194 real f9(real x) {
195     return 1.0 / (5 + x) + 1.0 / x;
196 }
197 
198 int main(void) {
199     int n = 15, population = 25, generation = 100000000, seed = 1;
200     clock_t time0 = 0, time1 = 0;
201     real runtime = 0;
202     real ftolerance = 1.0e-15, xtolerance = 1.0e-15;
203     real a = 0, f = 0, t = 0, x = 0, y = 0;
204     t = NewtonRaphson1(
205         0.5, ftolerance, xtolerance, f1, g);
206     printf_s("Solution of f(t) = cos(2t) + cos(3t)\n");
207     printf_s("t    = \t%14.12Lf\n", t);
208     printf_s("f(t) = \t%14.8e\n", f1(t));
```

```
209     x = NewtonRaphson2(
210         1.5, ftolerance, xtolerance, f2, CentralDifference);
211     printf_s("Solution of f(x) = sqrt(1 + sqrt(1 + x)) - x^1/3\n");
212     printf_s("x    = \t%14.12Lf\n", x);
213     printf_s("f(x) = \t%14.8e\n", f2(x));
214     x = NewtonRaphson2(
215         1.0, ftolerance, xtolerance, f3, CentralDifference);
216     printf_s("Solution of f(x) = 9^x + 12^x - 16^x\n");
217     printf_s("x    = \t%14.12Lf\n", x);
218     printf_s("f(x) = \t  %14.8e\n", f3(x));
219     x = NewtonRaphson2(
220         1.0, ftolerance, xtolerance, f4, CentralDifference);
221     printf_s("Solution of f(x) = 8^x-2^x - 2(6^x-3^x)\n");
222     printf_s("x    = \t%14.12Lf\n", x);
223     printf_s("f(x) = \t%14.8e\n", f4(x));
224     printf_s("Solution of f(a,x)=sin(sqrt(ax-x^2))=0\n");
225     printf_s("Subject to the constraint x+y=100\n");
226     printf_s("Where x and y are the two roots of\n");
227     printf_s("g(a,x)=ax-x^2-n*n*pi*pi=0\n");
228     printf_s("and n=%d\n", n);
229     time0 = clock();
230     HillClimber(n, population, generation, seed, &a, &f, &x, &y);
231     time1 = clock();
232     runtime = ((real)time1 - time0) / CLOCKS_PER_SEC;
233     printf_s("a = %15.12Lf\n", a);
234     printf_s("x = %15.12Lf\n", x);
235     printf_s("y = %15.12Lf\n", y);
236     printf_s("g = %15.12Lf\n", f);
237     printf_s("s = %15.12Lf\n", x + y);
238     printf_s("runtime in seconds = %Lf\n", runtime);
239     printf_s("Solve for a real root of the equation\n");
240     printf_s("f(x)=log6l(5+x)+log6l(x)=0\n");
241     printf_s("First we test our log6l(x) function\n");
242     printf_s("log6l(12) = %Lf\n", log6l(12));
243     printf_s("log6l(36) = %Lf\n", log6l(36));
244     x = NewtonRaphson1(
245         0.5, ftolerance, xtolerance, f8, f9);
246     printf_s("x = %12.10Lf\n", x);
247     printf_s("f = %12.10Lf\n", f8(x));
248     return 0;
249 }
```