```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
 */

#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)) {
```

```
53      return (fx(x + h) - fx(x - h)) / (2.0 * h);
54  }
55
56  real NewtonRaphson1(
57      real x0, real ftolerance, real xtolerence,
58      real(*fx)(real), real(*deriv)(real)) {
59      real x1 = 0.0;
60      while (1) {
61          real f = fx(x0);
62          real dfdx = deriv(x0);
63          x1 = x0 - f / dfdx;
64          if (fabsl(x1 - x0) < xtolerence ||
65              fabsl(f) < ftolerance)
66              break;
67          x0 = x1;
68      }
69      return x1;
70  }
71
72  real NewtonRaphson2(
73      real x0, real ftolerance, real xtolerence,
74      real(*fx)(real),
75      real(*deriv)(real, real, real(*fx)(real))) {
76      real h = 2.0 / 10000.0, x1 = 0.0;
77      while (1) {
78          real f = fx(x0);
79          real dfdx = deriv(x0, h, fx);
80          x1 = x0 - f / dfdx;
81          if (fabsl(x1 - x0) < xtolerence ||
82              fabsl(f) < ftolerance)
83              break;
84          x0 = x1;
85      }
86      return x1;
87  }
88
89  void QuadraticEquation(
90      real a, real b, real c,
91      real* x, real* y) {
92      real d = b * b - 4 * a * c;
93      if (d >= 0) {
94          *x = (-b + sqrtl(d)) / (2.0 * a);
95          *y = (-b - sqrtl(d)) / (2.0 * a);
96      }
97      else
98          *x = *y = 0.0;
99  }
100
101 real f5(real a, real x) {
102     return sinl(sqrtl(a * x - x * x));
103 }
104
```

```
105 real f6(int n, real a, real x) {
106     real pi = 4.0 * atanl(1.0);
107     return a * x - x * x - (real)n * n * pi * pi;
108 }
109
110 real f7(int n, real a, real x) {
111     return sqrt(f6(n, a, x));
112 }
113
114 void HillClimber(
115     int n, int population, int generation,
116     int seed, real* a, real* f, real* x,
117     real* y) {
118     int g = 0, i = 0, j = 0;
119     int* index = (int*)calloc(
120         population, sizeof(int));
121     real bestFitness = 1.0e16;
122     real childa = 0, childx = 0, childy = 0;
123     real pi = 4.0 * atanl(1.0);
124     real* fitness = (real*)calloc(
125         population, sizeof(real));
126     real** gene = (real**)calloc(
127         population, sizeof(real*));
128     if (fitness == NULL)
129         exit(-1);
130     if (gene == NULL)
131         exit(-2);
132     for (i = 0; i < population; i++) {
133         gene[i] = (real*)calloc(3, sizeof(real));
134         if (gene[i] == NULL)
135             exit(-3);
136     }
137     for (i = 0; i < population; i++) {
138         childa = gene[i][0] = 112.0 * (real)rand() / RAND_MAX;
139         QuadraticEquation(
140             -1.0, childa, -(real)n * n * pi * pi,
141             &childx, &childy);
142         fitness[i] = f6(n, childa, childx);
143         gene[i][1] = childx;
144         gene[i][2] = childy;
145         *a = childa;
146         *x = childx;
147         *y = childy;
148     }
149     for (g = 1; g <= generation; g++) {
150         int count = 0;
151         int parent1 = rand() % population;
152         int parent2 = rand() % population;
153         int parentc = parent1 < parent2 ?
154             parent1 : parent2;
155         real maximum = fitness[0];
156         childa = gene[parentc][0] + 0.0001;
```

```
157         QuadraticEquation(
158             -1.0, childa, -(real)n * n * pi * pi,
159             &childx, &childy);
160         real childf = f6(n, childa, childx);
161         for (i = 1; i < population; i++)
162             if (fitness[i] > maximum)
163                 maximum = fitness[i];
164         for (i = 0; i < population; i++)
165             if (fitness[i] == maximum)
166                 index[count++] = i;
167         if (childf < fitness[index[rand() % count]]) {
168             gene[parentc][0] = childa;
169             gene[parentc][1] = childx;
170             gene[parentc][2] = childy;
171             fitness[parentc] = childf;
172         }
173         if (childf < bestFitness &&
174             childf >= 0.0) {
175             *a = childa;
176             *x = childx;
177             *y = childy;
178             *f = childf;
179             bestFitness = childf;
180         }
181     }
182 }
183
184 int main(void) {
185     int n = 15, population = 25, generation = 100000000, seed = 1;
186     clock_t time0 = 0, time1 = 0;
187     real runtime = 0;
188     real ftolerance = 1.0e-15, xtolerance = 1.0e-15;
189     real a = 0, f = 0, t = 0, x = 0, y = 0;
190     t = NewtonRaphson1(
191         0.5, ftolerance, xtolerance, f1, g);
192     printf_s("Solution of f(t) = cos(2t) + cos(3t)\n");
193     printf_s("t    = \t%14.12Lf\n", t);
194     printf_s("f(t) = \t%14.8e\n", f1(t));
195     x = NewtonRaphson2(
196         1.5, ftolerance, xtolerance, f2, CentralDifference);
197     printf_s("Solution of f(x) = sqrt(1 + sqrt(1 + x)) - x^1/3\n");
198     printf_s("x    = \t%14.12Lf\n", x);
199     printf_s("f(x) = \t%14.8e\n", f2(x));
200     x = NewtonRaphson2(
201         1.0, ftolerance, xtolerance, f3, CentralDifference);
202     printf_s("Solution of f(x) = 9^x + 12^x - 16^x\n");
203     printf_s("x    = \t%14.12Lf\n", x);
204     printf_s("f(x) = \t  %14.8e\n", f3(x));
205     x = NewtonRaphson2(
206         1.0, ftolerance, xtolerance, f4, CentralDifference);
207     printf_s("Solution of f(x) = 8^x-2^x - 2(6^x-3^x)\n");
208     printf_s("x    = \t%14.12Lf\n", x);
```

```
209     printf_s("f(x) = \t%14.8e\n", f4(x));
210     printf_s("Solution of f(a,x)=sin(sqrt(ax-x^2))=0\n");
211     printf_s("Subject to the constraint x+y=100\n");
212     printf_s("Where x and y are the two roots of\n");
213     printf_s("g(a,x)=ax-x^2-n*n*pi*pi=0\n");
214     printf_s("and n=%d\n", n);
215     time0 = clock();
216     HillClimber(n, population, generation, seed, &a, &f, &x, &y);
217     time1 = clock();
218     runtime = ((real)time1 - time0) / CLOCKS_PER_SEC;
219     printf_s("a = %15.12Lf\n", a);
220     printf_s("x = %15.12Lf\n", x);
221     printf_s("y = %15.12Lf\n", y);
222     printf_s("g = %15.12Lf\n", f);
223     printf_s("s = %15.12Lf\n", x + y);
224     printf_s("runtime in seconds = %Lf\n", runtime);
225     return 0;
226 }
```