```cpp
#include <string>
#include <iostream>
#include <fstream>
#include <cmath>
#include <array>
#include <map>

//Constant expressions appearing in the problem
constexpr size_t dimension = 2;   //dimension of the reduced 1st-order problem
constexpr double PI = 3.14159265359;    //value of PI

//Definition of data types in the problem
typedef std::array<double, dimension> state_type;  //data type definition for dependant variables - array of x_0, x_1, ... x_n
typedef std::map<double, state_type> solution;   //data type definition for storing the list of calculated values ((hash)map of time -> state)

//Overload the + operator to be able to add two vectors
state_type operator + (state_type const& x, state_type const& y) {
    state_type z;
    for (size_t i = 0; i < dimension; i++) {
        z[i] = x[i] + y[i]; //add the individual components and store in z
    }
    return z;   //return the resulting vector z
}

//Overload the * operator to be able to multiply numbers and vectors
state_type operator * (double const& a, state_type const& x) {
    state_type z;
    for (size_t i = 0; i < dimension; i++) {
        z[i] = a * x[i];    //multiply the individual components and store in z
    }
    return z;   //return the resulting vector z
}

//This is the differential Equation, reduced to first-order
void Pendulum(const state_type& x, const double& t, state_type& dxdt) {
    dxdt[0] = x[1];
    dxdt[1] = -PI * PI * sin(x[0]);
}

//The stepper function, iteratively calculates x_{n+1} given the differential equation, x_{n} and step size
void rk4_step(void (*Diff_Equation)(const state_type& x, const double& t, state_type& dxdt), state_type& x, const double& t, const double& dt) {
    //temporary variables for intermediate steps
    state_type k1, k2, k3, k4;
```

```
44
45      //calculate the intermediate values
46      Diff_Equation(x, t, k1);    //calculate k1
47      Diff_Equation(x + (dt / 2.0) * k1, t + dt / 2.0, k2);   //calculate k2
48      Diff_Equation(x + (dt / 2.0) * k2, t + dt / 2.0, k3);   //calculate k3
49      Diff_Equation(x + dt * k3, t + dt, k4); //calculate k4
50
51      //calculate x_{n+1} using the RK4 formula and return the results
52      x = x + (dt / 6.0) * (k1 + 2 * k2 + 2 * k3 + k4);
53  }
54
55  int main() {
56      solution x_t_0, x_t_1, x_t_12;   //variable to store the calculations
57
58      size_t STEPS = 1024;  //number of steps
59      double t_0 = 0.0;   //initial time
60      double t_1 = 1.0;   //final time
61      double dt = (t_1 - t_0) / (STEPS - 1); //step size
62
63      size_t iteration = 0;
64      double left = 0, right = 10, middle = 5;
65
66      while (iteration < 100) {
67          state_type x = { 0.0, left };   //initial values for dependant
              variables
68          //Step through the domain of the problem and store the solutions
69          x_t_0[t_0] = x;   //store initial values
70          for (size_t i = 0; i < STEPS; i++) {
71              rk4_step(Pendulum, x, NULL, dt);    //step forward
72              x_t_0[t_0 + i * dt] = x;  //store the calculation
73          }
74          x = { 0.0, right };   //initial values for dependant variables
75          x_t_1[t_0] = x;   //store initial values
76          for (size_t i = 0; i < STEPS; i++) {
77              rk4_step(Pendulum, x, NULL, dt);    //step forward
78              x_t_1[t_0 + i * dt] = x;  //store the calculation
79          }
80          middle = (left + right) / 2.0;
81          x = { 0.0, middle };   //initial values for dependant variables
82          x_t_12[t_0] = x;   //store initial values
83          for (size_t i = 0; i < STEPS; i++) {
84              rk4_step(Pendulum, x, NULL, dt);    //step forward
85              x_t_12[t_0 + i * dt] = x;  //store the calculation
86          }
87          double l = x_t_0[1][0] - PI / 4, r = x_t_1[1][0] - PI / 4, m =
              x_t_12[1][0] - PI / 4;
88          if ((l > 0 && r > 0) || (l < 0 && r < 0)) {
89              return -1;
90          }
```

```
91
92          if ((l < 0 && m < 0) || (l > 0 && m > 0)) {
93              left = middle;
94          }
95          else if ((r < 0 && m < 0) || (r > 0 && m > 0)) {
96              right = middle;
97          }
98          else {
99              return -1;
100         }
101
102         std::cout << iteration << "\t" << left << "\t" << l << "\t" <<
              right << "\t" << r << std::endl;
103         iteration++;
104     }
105
106
107
108     std::ofstream outfile;  //file handle to save the results in a file
109     outfile.open("./output/BVP.txt", std::ios::out | std::ios::trunc);
110     for (auto const& temp : x_t_12) {
111         outfile << temp.first << "\t" << temp.second[0] << "\t" <<
              temp.second[1] << std::endl;
112     }
113     outfile.close();
114 }
```