

```
1 #include <iostream>
2 #include <fstream>
3 #include <vector>
4 #include <complex>
5
6 //Types used in the problem
7 typedef std::vector<std::complex<double>> Complexlist;
8
9 //Constant expressions appearing in the problem
10 constexpr double pi = 3.14159265359;
11 constexpr std::complex<double> i(0, 1);
12
13 //This is the function to be taken DFT of
14 std::complex<double> function(double t) {
15     return exp(2.0 * pi * i * t - t);
16 }
17
18 //Helper function to discretize the given function
19 Complexlist discretizer(std::complex<double>(*function)(double t), double starting_point, double ending_point, size_t STEPS) {
20     Complexlist result(STEPS);
21     for (size_t i = 0; i < STEPS; i++) {
22         result[i] = function(starting_point + i * (ending_point - starting_point) / (STEPS - 1));
23     }
24
25     return result;
26 }
27
28 //FFT algorithm
29 Complexlist FFT(Complexlist P) {
30     size_t n = P.size();
31
32     if (n == 1) {
33         return P;
34     }
35
36     //n roots of unity
37     std::complex<double> w = std::exp(-2.0 * pi * i / (double)n);
38
39     Complexlist Pe, Po;
40
41     //Divide into odd and even polynomials
42     for (size_t j = 0; j < n / 2; j++) {
43         Pe.push_back(P[2 * j]);
44         Po.push_back(P[2 * j + 1]);
45     }
46
47     //invoke recursion
```

```
48     Complexlist ye = FFT(Pe), yo = FFT(Po), y(n);
49
50     //tie up everything to complete
51     for (size_t j = 0; j < n / 2; j++) {
52         y[j] = ye[j] + pow(w, j) * yo[j];
53         y[j + n / 2] = ye[j] - pow(w, j) * yo[j];
54     }
55
56     return y;
57 }
58
59 int main() {
60     //Discretize given function in 2048 steps
61     Complexlist coefficients = discretizer(function, 0, 8, 2048);
62
63     //Calculate the FFT
64     Complexlist values = FFT(coefficients);
65
66     std::ofstream outfile; //file handle to save the results in a file
67     outfile.open("./output/FFT.txt", std::ios::out | std::ios::trunc);
68
69     for (auto& value : values) { //Loop through the array to store the values
70         outfile << value.real() << "\t" << value.imag() << std::endl; // write to the output file
71     }
72     outfile.close(); //when done, close the file.
73
74     return 0;
75 }
```

















