Uploaded on Github as well for your reference

https://github.com/Roshen98/Interpolation_Polynomial.git

Consider the following data points:

(3,1), (1,0.12), (0,-0.3),(4,2),(7,2.5)

Exercise 1

Construct an interpolation polynomial using the Lagrange's Interpolation method, and evaluate at x=2.

$(3,1),\ (1,0.12),\ (0,-0.3),\ (4,2),\ (7,2.5)$

Compute Basis Polynomial

$$L_0(x) = \frac{(x-1)}{(3-1)} \cdot \frac{(x-0)}{(3-0)} \cdot \frac{(x-4)}{(3-4)} \cdot \frac{(x-7)}{(3-7)}$$

$$= \frac{1}{2}(x-1)\left(\frac{x}{3}\right)(-1)(x-4)(-1/4)(x-7)$$

$$= \frac{1}{24}(x-1)(x)(x-4)(x-7) \quad \text{degree : } 4$$

$$L_1(x) = \frac{(x-3)}{(1-3)} \cdot \frac{(x-0)}{(1-0)} \cdot \frac{(x-4)}{(1-4)} \cdot \frac{(x-7)}{(1-7)}$$

$$= -\frac{1}{36}(x-3)(x+0)(x-4)(x-7)$$

$$L_2(x) = \frac{(x-3)}{(0-3)} \cdot \frac{(x-1)}{(0-1)} \cdot \frac{(x-4)}{(0-4)} \cdot \frac{(x-7)}{(0-7)}$$

$$= \frac{1}{84}(x-3)(x-1)(x-4)(x-7)$$

$$L_3(x) = \frac{(x-3)}{(4-3)} \cdot \frac{(x-1)}{(4-1)} \cdot \frac{(x-0)}{(4-0)} \cdot \frac{(x-7)}{(4-7)}$$

$$= -\frac{1}{36}(x-3)(x-1)(x)(x-7)$$

$$L_4(x) = \frac{(x-3)}{(7-3)} \cdot \frac{(x-1)}{(7-1)} \cdot \frac{(x-0)}{(7-0)} \cdot \frac{(x-4)}{\cancel{x}(7-4)}$$

$$= \frac{1}{504}(x-3)(x-1)(x)(x-4)$$

Build the Polynomial:

$$L(x) = \left[\tfrac{1}{24}(x-1)(x)(x-4)(x-7)\right]\cdot 1 + \left[\overset{1/36}{\cancel{\tfrac{1}{24}}}(x-3)(x)(x-4)(x-7)\right]\cdot 0.12$$

$$+\left[\frac{1}{84}(x-3)(x-1)(\overset{x-4}{\cancel{x}})(x-7)\right]\cdot -0.3 + \left[-\frac{1}{36}(x-3)(x-1)(x)(x-7)\right]\cdot 2 +$$

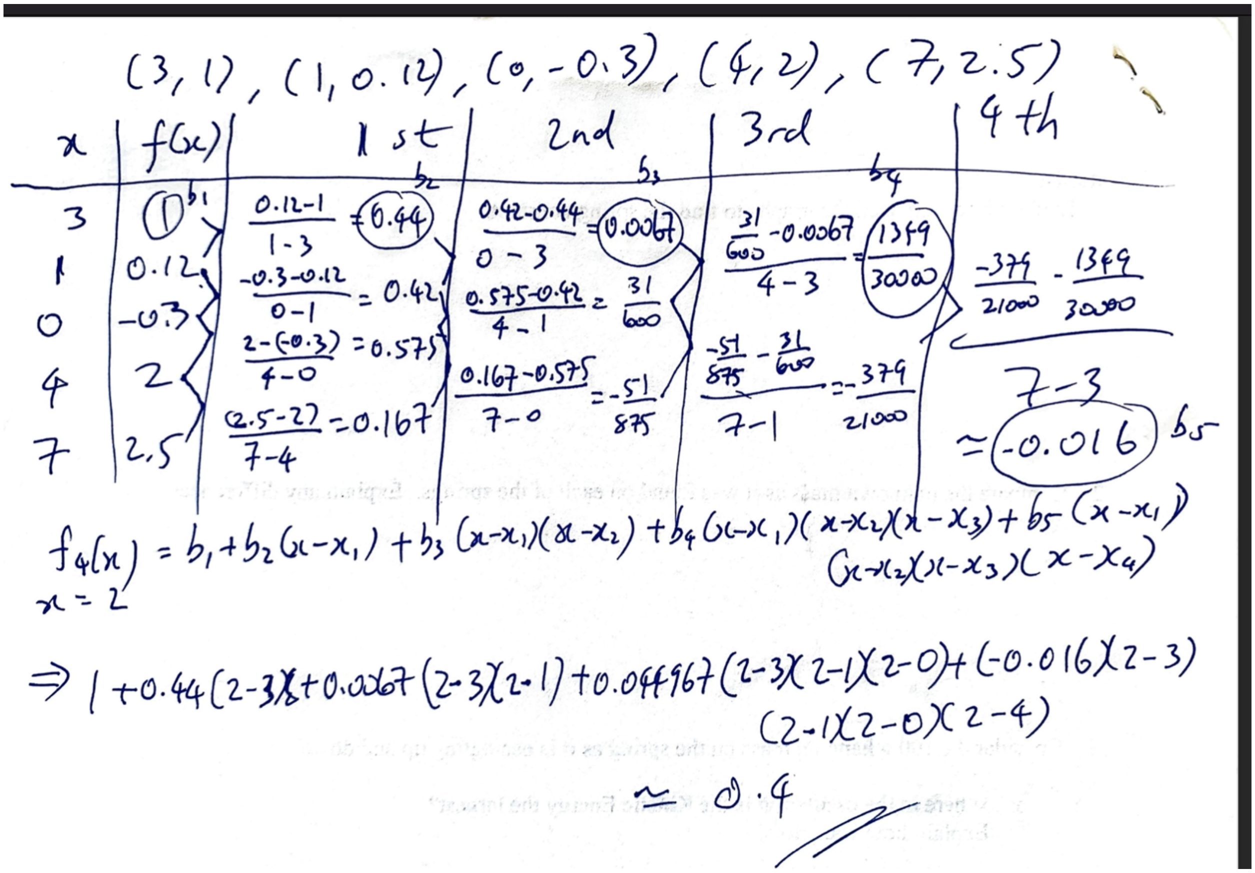
$$\left[\tfrac{1}{504}(x-3)(x-1)(x)(x-4)\right]\cdot 2.5,$$

Plug in $x = 2$

$L(x) \Rightarrow 0.4$

Exercise 2

Construct an interpolation polynomial using the Newton's Interpolation method, and evaluate at x=2.



$(3, 1), (1, 0.12), (0, -0.3), (4, 2), (7, 2.5)$

x	$f(x)$	1st	2nd	3rd	4th
3	$\boxed{1}\ b_1$	b_2: $\frac{0.12-1}{1-3} = \boxed{0.44}$	b_3: $\frac{0.42-0.44}{0-3} = \boxed{0.0067}$	b_4: $\frac{\frac{31}{600}-0.0067}{4-3} = \boxed{\frac{1349}{30000}}$	$\frac{\frac{-379}{21000}-\frac{1349}{30000}}{7-3} \simeq \boxed{-0.016}\ b_5$
1	0.12	$\frac{-0.3-0.12}{0-1} = 0.42$	$\frac{0.575-0.42}{4-1} = \frac{31}{600}$	$\frac{\frac{-51}{875}-\frac{31}{600}}{7-1} = -\frac{379}{21000}$	
0	-0.3	$\frac{2-(-0.3)}{4-0} = 0.575$	$\frac{0.167-0.575}{7-0} = -\frac{51}{875}$		
4	2	$\frac{(2.5-2)}{7-4} = 0.167$			
7	2.5				

$$f_4(x) = b_1 + b_2(x-x_1) + b_3(x-x_1)(x-x_2) + b_4(x-x_1)(x-x_2)(x-x_3) + b_5(x-x_1)(x-x_2)(x-x_3)(x-x_4)$$

$x = 2$

$$\Rightarrow 1 + 0.44(2-3) + 0.0067(2-3)(2-1) + 0.044967(2-3)(2-1)(2-0) + (-0.016)(2-3)(2-1)(2-0)(2-4)$$

$$\simeq 0.4$$

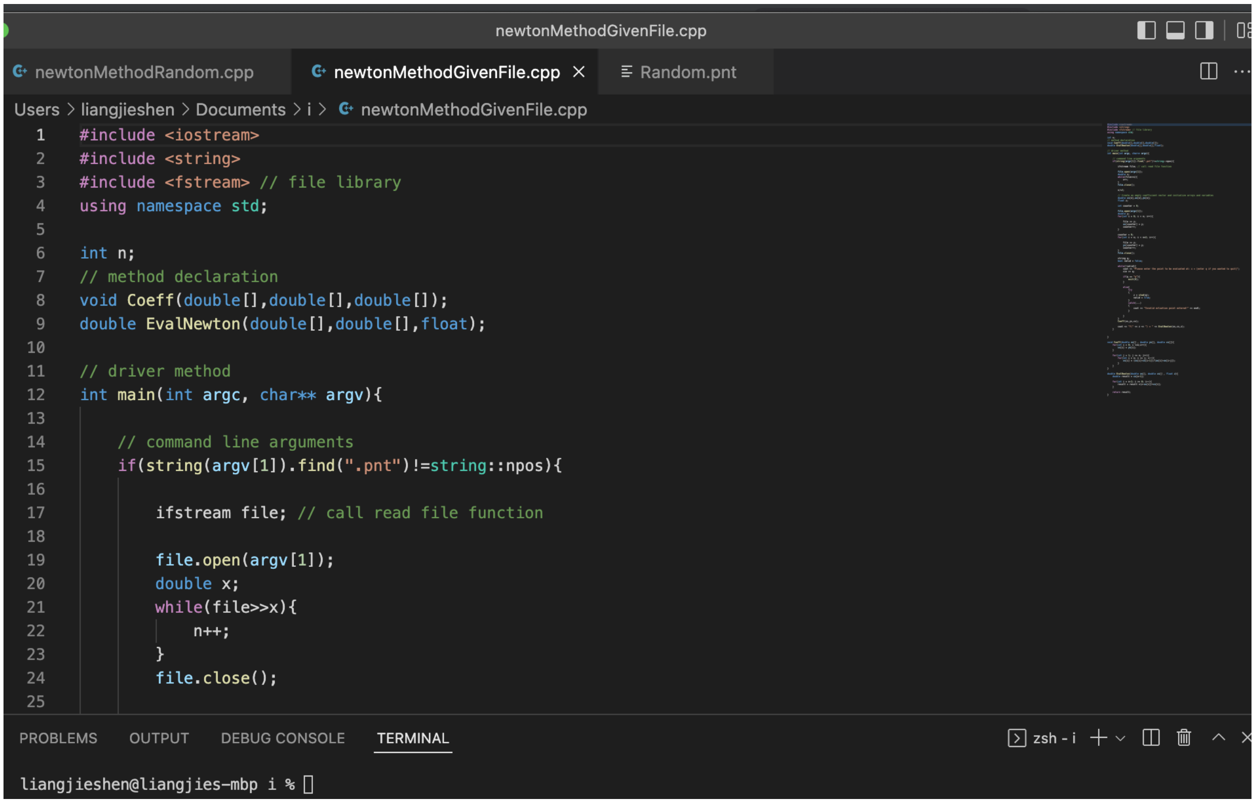
Exercise 3

Write a program, in the language of your preference, that is used to compute Newton's interpolation. The program should take as input a file that contains the data points in the following format:

x0 x1 x2 ... xn

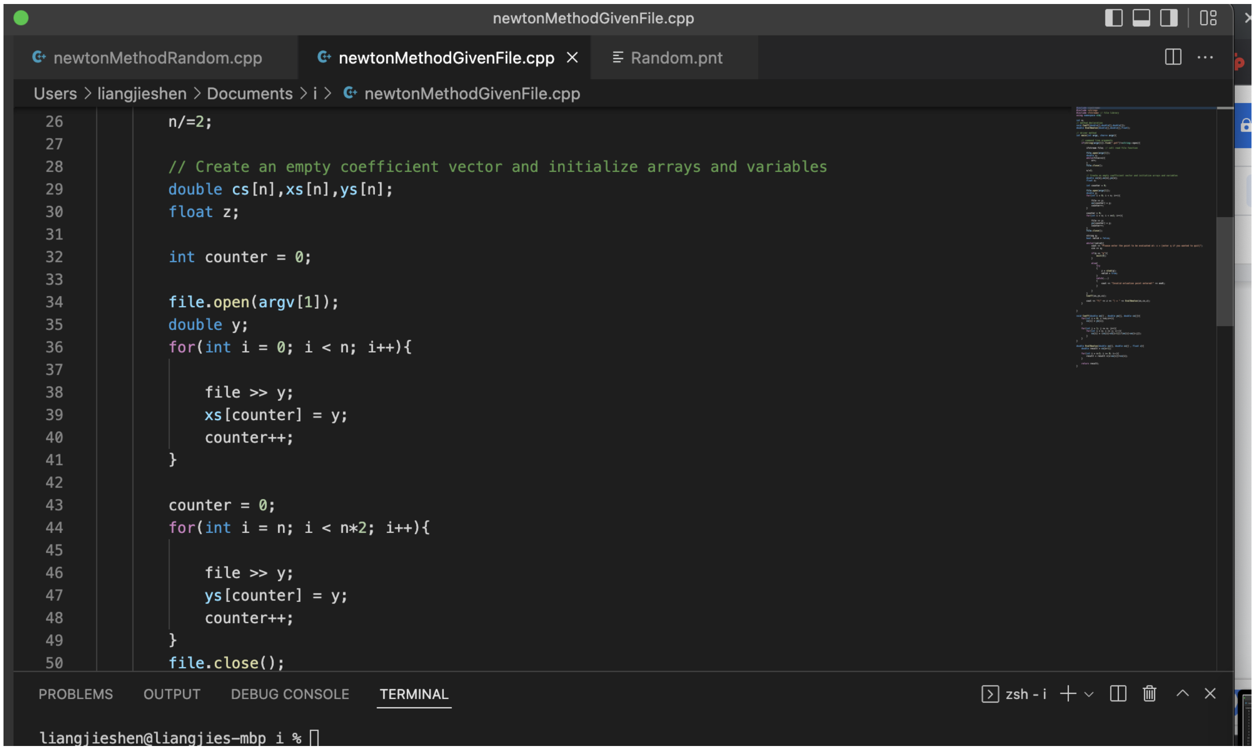
y0 y1 y2 ... yn

that is, the first rows contains the first element of each data point, and the next row the second one. After processing the input, the program will provide a prompt asking of a value to be used to evaluate the polynomial and print the result of such evaluation, going back to the prompt. Entering 'q' instead of a real number will exit the program. Use your program to with the example in the exercises above to check your solution.


newtonMethodGivenFile.cpp
newtonMethodRandom.cpp
newtonMethodGivenFile.cpp
Random.pnt
Users > liangjieshen > Documents > i > newtonMethodGivenFile.cpp
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
zsh - i

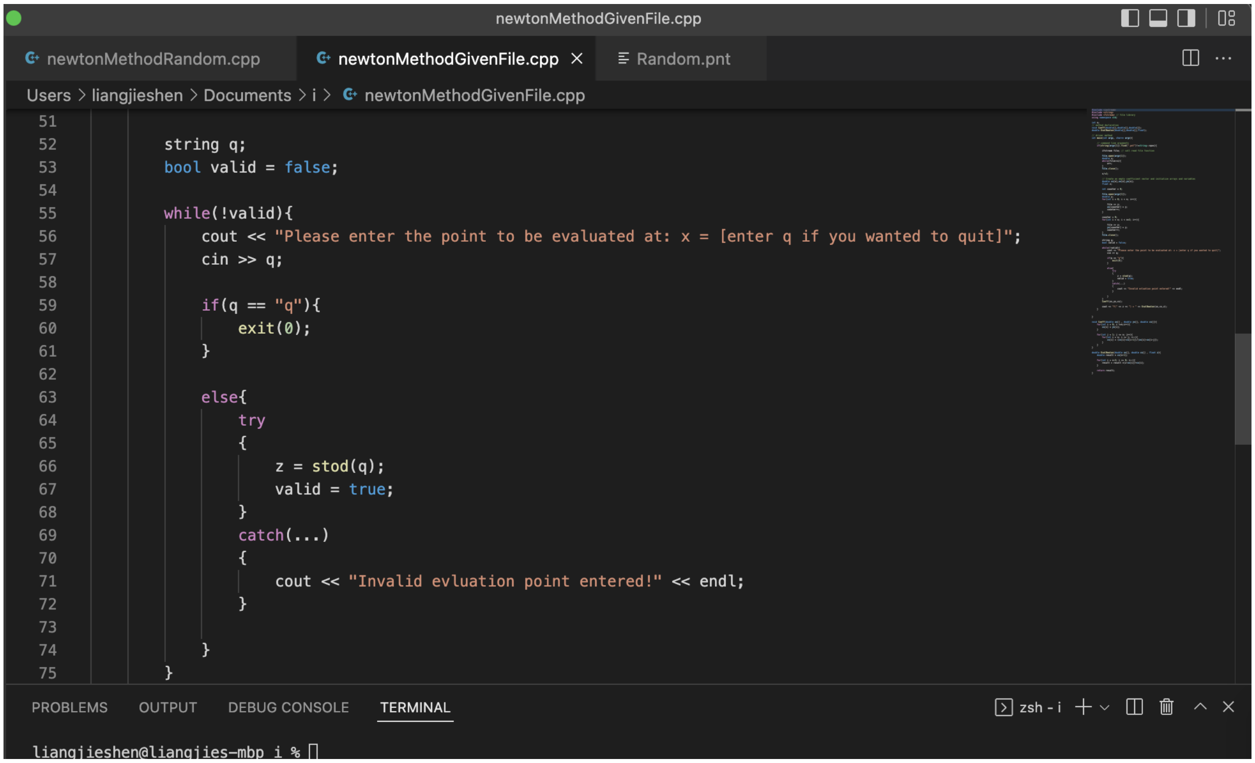
```cpp
#include <iostream>
#include <string>
#include <fstream> // file library
using namespace std;

int n;
// method declaration
void Coeff(double[],double[],double[]);
double EvalNewton(double[],double[],float);

// driver method
int main(int argc, char** argv){

    // command line arguments
    if(string(argv[1]).find(".pnt")!=string::npos){

        ifstream file; // call read file function

        file.open(argv[1]);
        double x;
        while(file>>x){
            n++;
        }
        file.close();

```

```
liangjieshen@liangjies-mbp i %
```


newtonMethodGivenFile.cpp
newtonMethodRandom.cpp
newtonMethodGivenFile.cpp
Random.pnt
Users > liangjieshen > Documents > i > newtonMethodGivenFile.cpp
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
zsh - i

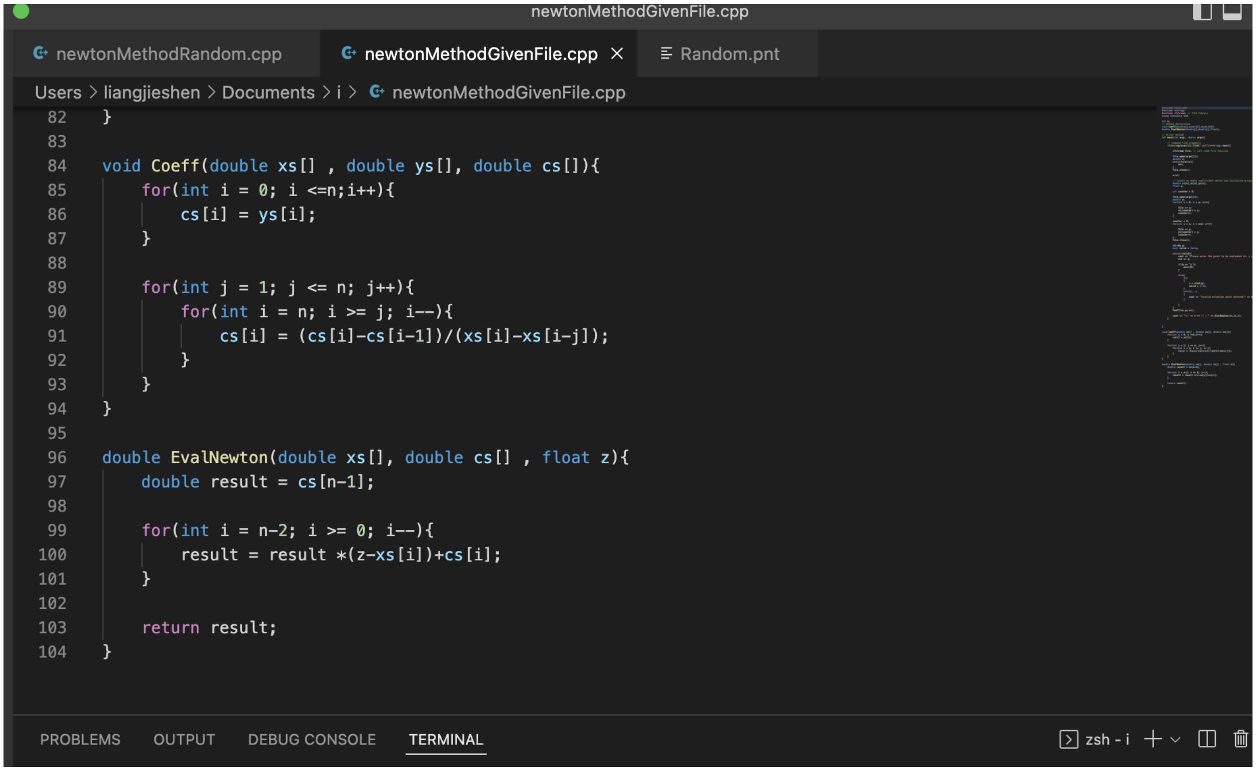
```
        n/=2;

        // Create an empty coefficient vector and initialize arrays and variables
        double cs[n],xs[n],ys[n];
        float z;

        int counter = 0;

        file.open(argv[1]);
        double y;
        for(int i = 0; i < n; i++){

            file >> y;
            xs[counter] = y;
            counter++;
        }

        counter = 0;
        for(int i = n; i < n*2; i++){

            file >> y;
            ys[counter] = y;
            counter++;
        }
        file.close();
```

```
liangjieshen@liangjies-mbp i % 
```

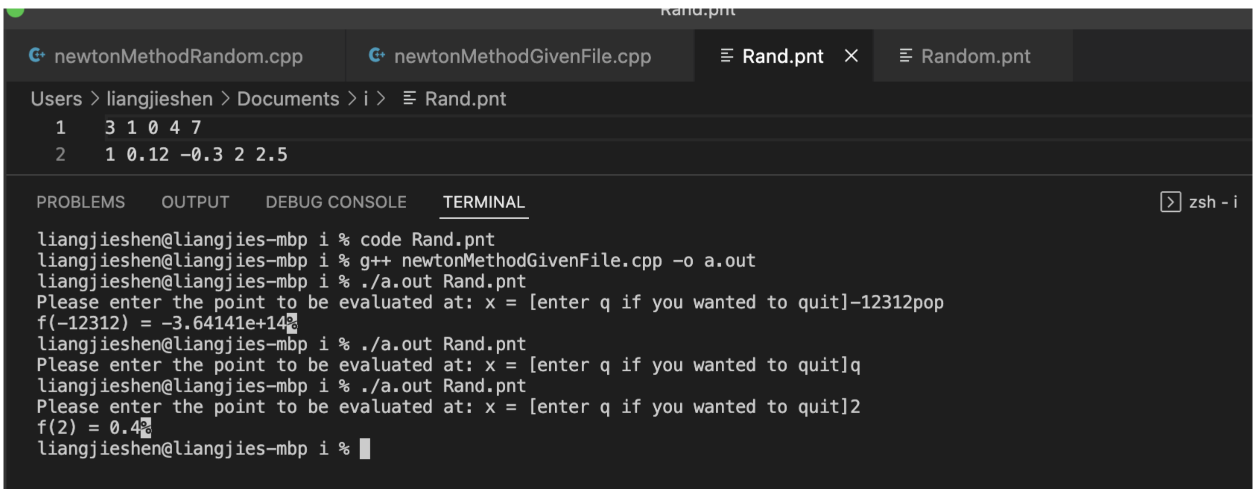

newtonMethodGivenFile.cpp
newtonMethodRandom.cpp
newtonMethodGivenFile.cpp
Random.pnt
Users > liangjieshen > Documents > i > newtonMethodGivenFile.cpp
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
zsh - i

```

        string q;
        bool valid = false;

        while(!valid){
            cout << "Please enter the point to be evaluated at: x = [enter q if you wanted to quit]";
            cin >> q;

            if(q == "q"){
                exit(0);
            }

            else{
                try
                {
                    z = stod(q);
                    valid = true;
                }
                catch(...)
                {
                    cout << "Invalid evluation point entered!" << endl;
                }

            }
        }
```

```
liangjieshen@liangjies-mbp i % 
```


newtonMethodGivenFile.cpp
newtonMethodRandom.cpp
newtonMethodGivenFile.cpp
Random.pnt
Users > liangjieshen > Documents > i > newtonMethodGivenFile.cpp
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
zsh - i

```
}

void Coeff(double xs[] , double ys[], double cs[]){
    for(int i = 0; i <=n;i++){
        cs[i] = ys[i];
    }

    for(int j = 1; j <= n; j++){
        for(int i = n; i >= j; i--){
            cs[i] = (cs[i]-cs[i-1])/(xs[i]-xs[i-j]);
        }
    }
}

double EvalNewton(double xs[], double cs[] , float z){
    double result = cs[n-1];

    for(int i = n-2; i >= 0; i--){
        result = result *(z-xs[i])+cs[i];
    }

    return result;
}
```



```
3 1 0 4 7
1 0.12 -0.3 2 2.5
```

```
liangjieshen@liangjies-mbp i % code Rand.pnt
liangjieshen@liangjies-mbp i % g++ newtonMethodGivenFile.cpp -o a.out
liangjieshen@liangjies-mbp i % ./a.out Rand.pnt
Please enter the point to be evaluated at: x = [enter q if you wanted to quit]-12312pop
f(-12312) = -3.64141e+14%
liangjieshen@liangjies-mbp i % ./a.out Rand.pnt
Please enter the point to be evaluated at: x = [enter q if you wanted to quit]q
liangjieshen@liangjies-mbp i % ./a.out Rand.pnt
Please enter the point to be evaluated at: x = [enter q if you wanted to quit]2
f(2) = 0.4%
liangjieshen@liangjies-mbp i % █
```

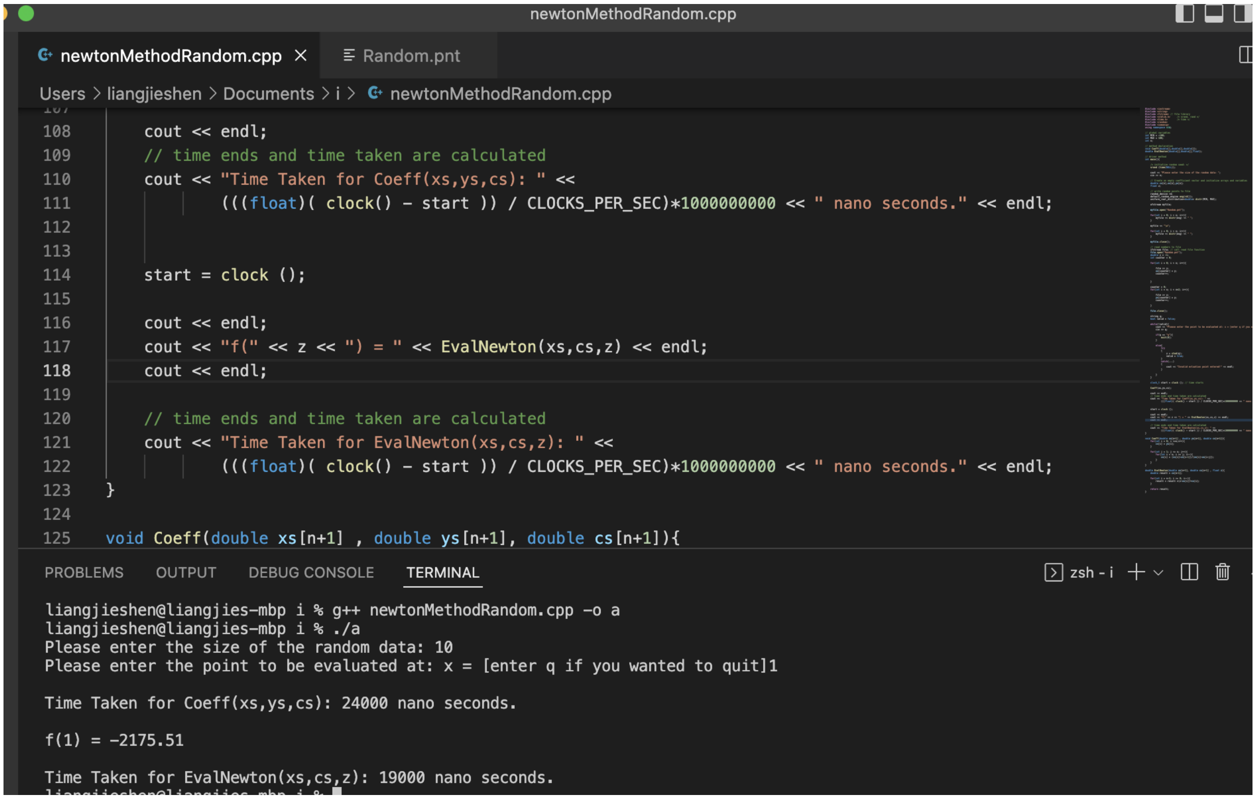
Exercise 4

Create a program that takes a positive integer 'n' as input the produces a random data set file with 'n' points, in the format of the previous exercise. Use this program to create data sets with 10, 100, and 1000 points and run your program with these data sets.

Evaluate at random points. Report on the time it takes to compute interpolation and evaluate in each instance.

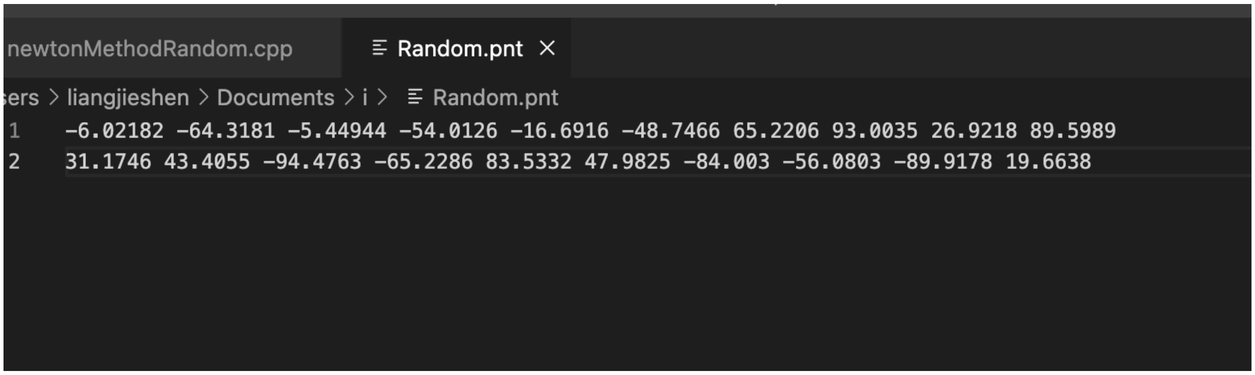
—

DATA FOR 10



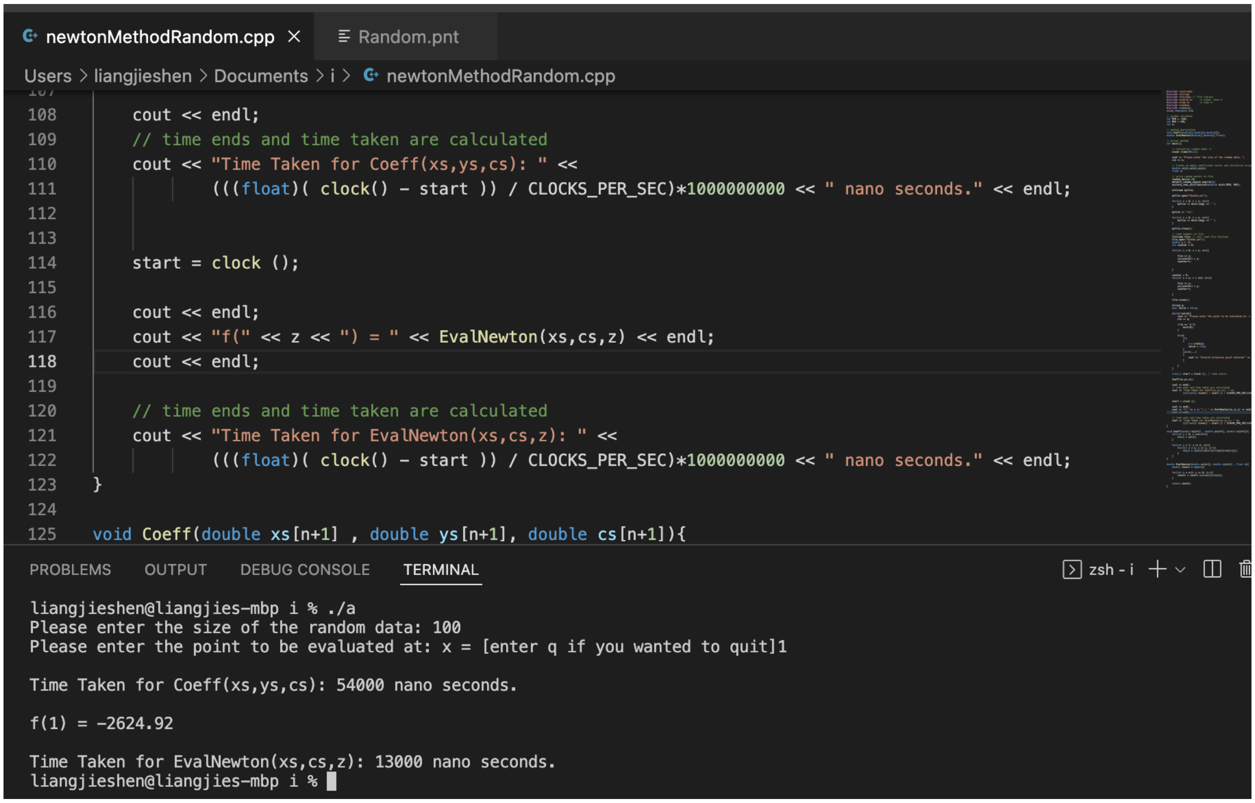
```
    cout << endl;
    // time ends and time taken are calculated
    cout << "Time Taken for Coeff(xs,ys,cs): " <<
        (((float)( clock() - start )) / CLOCKS_PER_SEC)*1000000000 << " nano seconds." << endl;


    start = clock ();

    cout << endl;
    cout << "f(" << z << ") = " << EvalNewton(xs,cs,z) << endl;
    cout << endl;

    // time ends and time taken are calculated
    cout << "Time Taken for EvalNewton(xs,cs,z): " <<
        (((float)( clock() - start )) / CLOCKS_PER_SEC)*1000000000 << " nano seconds." << endl;
}

void Coeff(double xs[n+1] , double ys[n+1], double cs[n+1]){
```

```
liangjieshen@liangjies-mbp i % g++ newtonMethodRandom.cpp -o a
liangjieshen@liangjies-mbp i % ./a
Please enter the size of the random data: 10
Please enter the point to be evaluated at: x = [enter q if you wanted to quit]1

Time Taken for Coeff(xs,ys,cs): 24000 nano seconds.

f(1) = -2175.51

Time Taken for EvalNewton(xs,cs,z): 19000 nano seconds.
liangjieshen@liangjies-mbp i %
```



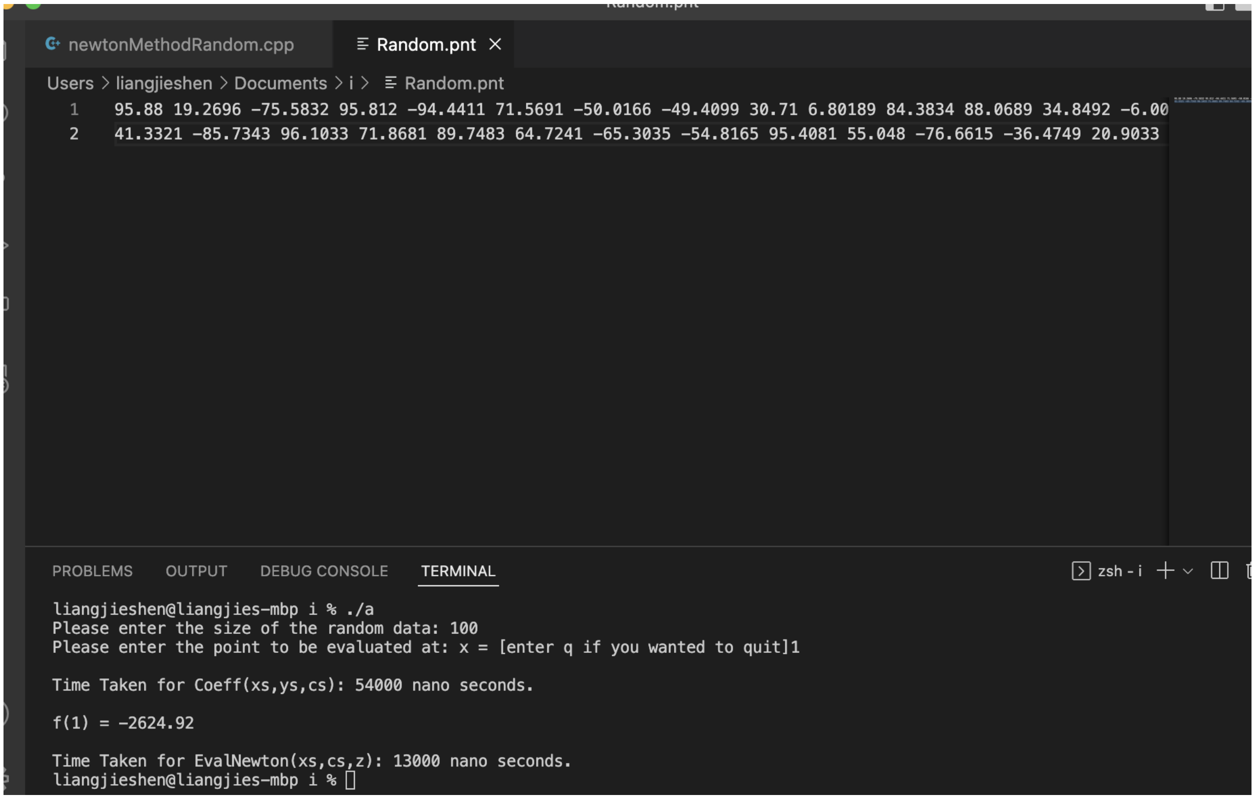
```
-6.02182 -64.3181 -5.44944 -54.0126 -16.6916 -48.7466 65.2206 93.0035 26.9218 89.5989
31.1746 43.4055 -94.4763 -65.2286 83.5332 47.9825 -84.003 -56.0803 -89.9178 19.6638
```

DATA FOR 100



```
    cout << endl;
    // time ends and time taken are calculated
    cout << "Time Taken for Coeff(xs,ys,cs): " <<
            (((float)( clock() - start )) / CLOCKS_PER_SEC)*1000000000 << " nano seconds." << endl;


    start = clock ();

    cout << endl;
    cout << "f(" << z << ") = " << EvalNewton(xs,cs,z) << endl;
    cout << endl;

    // time ends and time taken are calculated
    cout << "Time Taken for EvalNewton(xs,cs,z): " <<
            (((float)( clock() - start )) / CLOCKS_PER_SEC)*1000000000 << " nano seconds." << endl;
}

void Coeff(double xs[n+1] , double ys[n+1], double cs[n+1]){
```

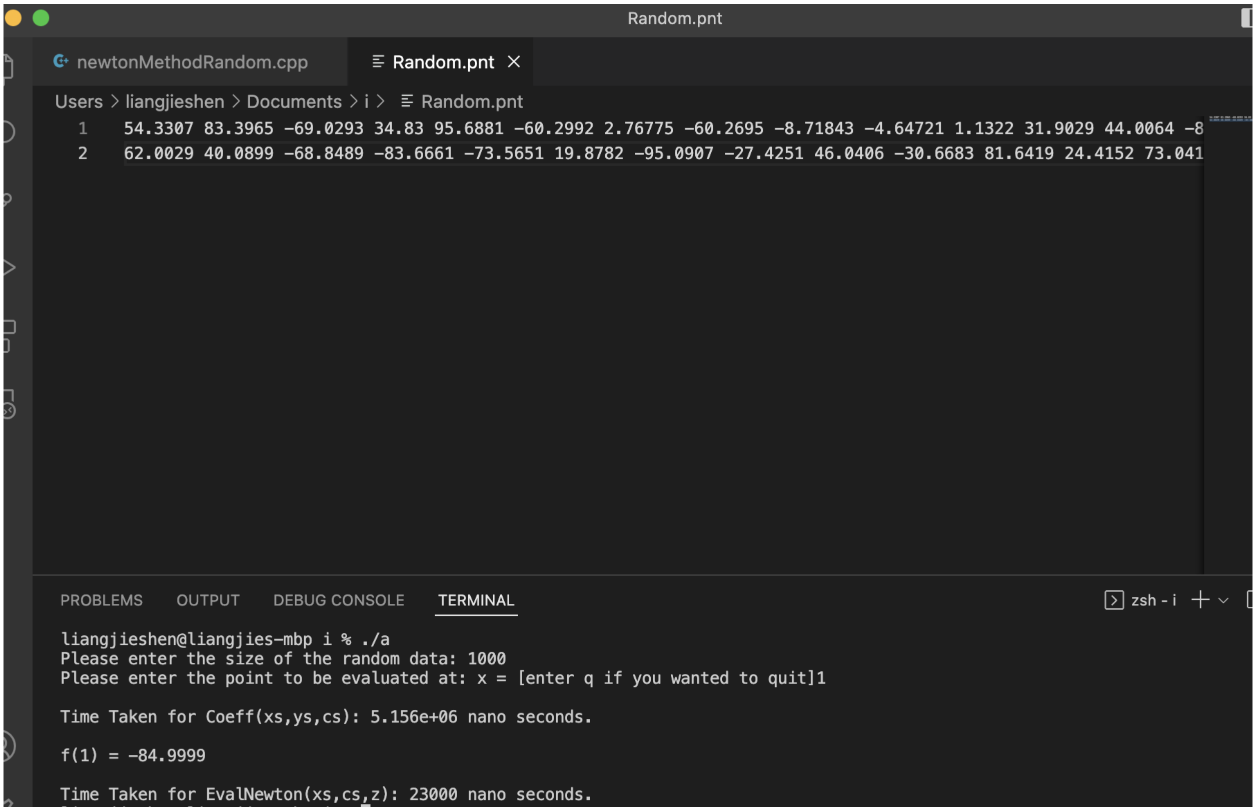
```
liangjieshen@liangjies-mbp i % ./a
Please enter the size of the random data: 100
Please enter the point to be evaluated at: x = [enter q if you wanted to quit]1

Time Taken for Coeff(xs,ys,cs): 54000 nano seconds.

f(1) = -2624.92

Time Taken for EvalNewton(xs,cs,z): 13000 nano seconds.
liangjieshen@liangjies-mbp i %
```



```
95.88 19.2696 -75.5832 95.812 -94.4411 71.5691 -50.0166 -49.4099 30.71 6.80189 84.3834 88.0689 34.8492 -6.00
41.3321 -85.7343 96.1033 71.8681 89.7483 64.7241 -65.3035 -54.8165 95.4081 55.048 -76.6615 -36.4749 20.9033
```

```
liangjieshen@liangjies-mbp i % ./a
Please enter the size of the random data: 100
Please enter the point to be evaluated at: x = [enter q if you wanted to quit]1

Time Taken for Coeff(xs,ys,cs): 54000 nano seconds.

f(1) = -2624.92

Time Taken for EvalNewton(xs,cs,z): 13000 nano seconds.
liangjieshen@liangjies-mbp i %
```

DATA FOR 1000



```
54.3307 83.3965 -69.0293 34.83 95.6881 -60.2992 2.76775 -60.2695 -8.71843 -4.64721 1.1322 31.9029 44.0064 -8
62.0029 40.0899 -68.8489 -83.6661 -73.5651 19.8782 -95.0907 -27.4251 46.0406 -30.6683 81.6419 24.4152 73.041
```

```
liangjieshen@liangjies-mbp i % ./a
Please enter the size of the random data: 1000
Please enter the point to be evaluated at: x = [enter q if you wanted to quit]1

Time Taken for Coeff(xs,ys,cs): 5.156e+06 nano seconds.

f(1) = -84.9999

Time Taken for EvalNewton(xs,cs,z): 23000 nano seconds.
```