Assignment

Data Analysis

Submitted by: Nishtha Chadha

Roll no.: UEM203128

**Code for the implementation of problem:**

#include <bits/stdc++.h>

using namespace std;

// Mean, Variance and Standard Deviation

void Problem1(vector<int> &data)

{

    double mean = accumulate(data.begin(), data.end(), 0.0) / data.size();

    double variance = 0.0;

    for (int i = 0; i < data.size(); i++)

        variance += (data[i] - mean) \* (data[i] - mean);

    variance /= data.size();

    double stdDev = sqrt(variance);

    cout << "Mean: " << mean << endl;

    cout << "Variance: " << variance << endl;

    cout << "Standard Deviation: " << stdDev << endl;

}

// Gradient and Laplacian

void Problem2(vector<int> &data)

{

    vector<int> gradient;

    for (int i = 0; i < data.size() - 1; i++)

        gradient.push\_back(data[i + 1] - data[i]);

    vector<int> laplacian;

    for (int i = 0; i < gradient.size() - 1; i++)

        laplacian.push\_back(gradient[i + 1] - gradient[i]);

    cout << "Gradient: ";

    for (int i = 0; i < gradient.size(); i++)

        cout << gradient[i] << " ";

    cout << endl;

    cout << "Laplacian: ";

    for (int i = 0; i < laplacian.size(); i++)

        cout << laplacian[i] << " ";

    cout << endl;

}

// Smoothening Filter of Size 5

void Problem3(vector<int> data)

{

    vector<int> smoothened;

    for (int i = 0; i < data.size(); i++)

    {

        int sum = 0;

        for (int j = i - 2; j <= i + 2; j++)

        {

            if (j < 0)

            {

                sum += data[0];

            }

            else if (j >= data.size())

            {

                sum += data[data.size() - 1];

            }

            else

            {

                sum += data[j];

            }

        }

        smoothened.push\_back(sum / 5);

    }

    cout << "Smoothened: ";

    for (int i = 0; i < smoothened.size(); i++)

        cout << smoothened[i] << " ";

    cout << endl;

}

// Moving average for different values of Beta

void Problem4(vector<int> &data)

{

    vector<double> movingAverage;

    for (double beta = 0.1; beta <= 1.0; beta += 0.1)

    {

        double movingAverageValue = 0.0;

        for (int i = 0; i < data.size(); i++)

        {

            movingAverageValue = beta \* data[i] + (1 - beta) \* movingAverageValue;

            movingAverage.push\_back(movingAverageValue);

        }

        cout << "Beta: " << beta << endl;

        cout << "Moving Average: ";

        for (int i = 0; i < movingAverage.size(); i++)

            cout << movingAverage[i] << " ";

        cout << endl;

    }

}

// Normalisation using Mean Score and Adam optimisation

void Problem5(vector<int> &data)

{

    double mean = accumulate(data.begin(), data.end(), 0.0) / data.size();

    int max = \*max\_element(data.begin(), data.end()),

        min = \*min\_element(data.begin(), data.end());

    vector<double> normalisedData;

    for (int i = 0; i < data.size(); i++)

        normalisedData.push\_back(abs(data[i] - mean) / (max - min));

    cout << "Normalised Data: ";

    for (int i = 0; i < normalisedData.size(); i++)

        cout << normalisedData[i] << " ";

    cout << endl;

    // Adam Optimisation

    vector<int> gradient;

    for (int i = 0; i < data.size() - 1; i++)

        gradient.push\_back(data[i + 1] - data[i]);

    vector<int> laplacian;

    for (int i = 0; i < gradient.size() - 1; i++)

        laplacian.push\_back(gradient[i + 1] - gradient[i]);

    vector<double> adamOptimised;

    for (int i = 0; i < laplacian.size(); i++)

    {

        // Took abs because of cases like sqrt(-1)

        adamOptimised.push\_back(data[i + 2] - 0.01 \* gradient[i + 1] / (sqrt(abs(laplacian[i])) + 0.1));

    }

    cout << "Adam Optimised: ";

    for (int i = 0; i < adamOptimised.size(); i++)

        cout << adamOptimised[i] << " ";

    cout << endl;

}

void runProblems(vector<int> &data)

{

    Problem1(data);

    Problem2(data);

    Problem3(data);

    Problem4(data);

    Problem5(data);

}

int main()

{

    int n;

    cin >> n;

    vector<int> arr(n);

    for (int &x : arr)

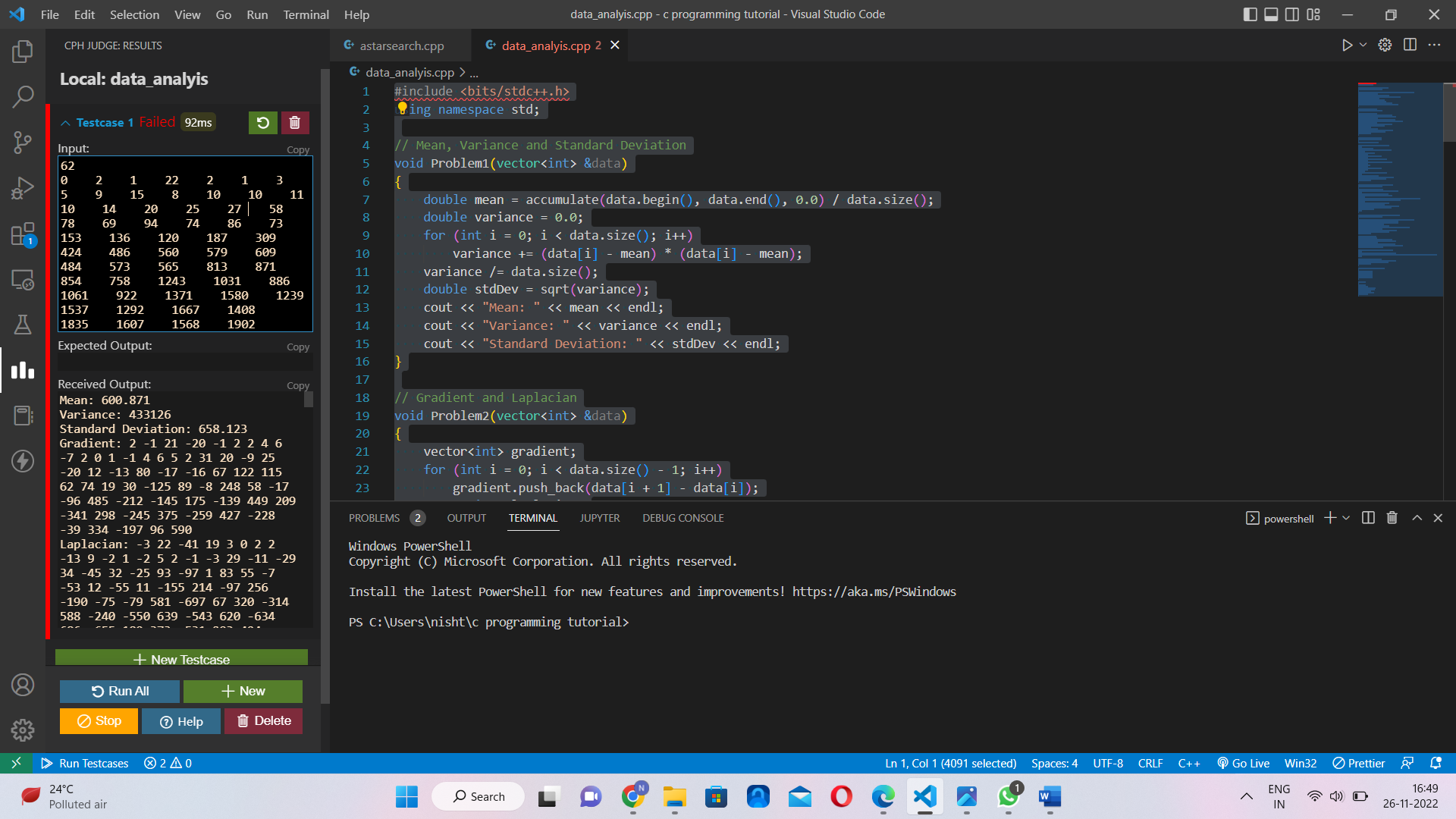
        cin >> x;

    runProblems(arr);

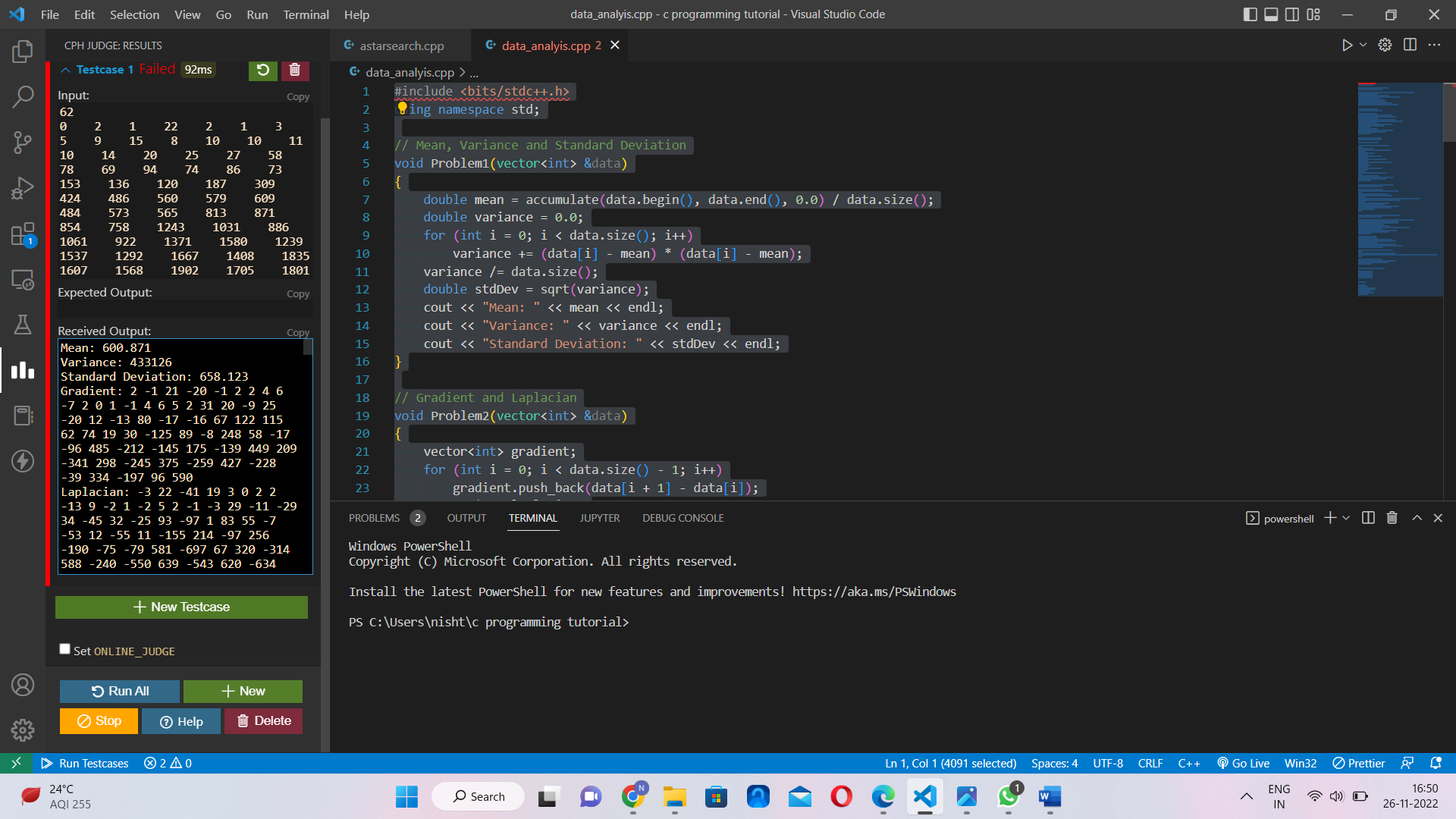
    return 0;

}

**Input:**

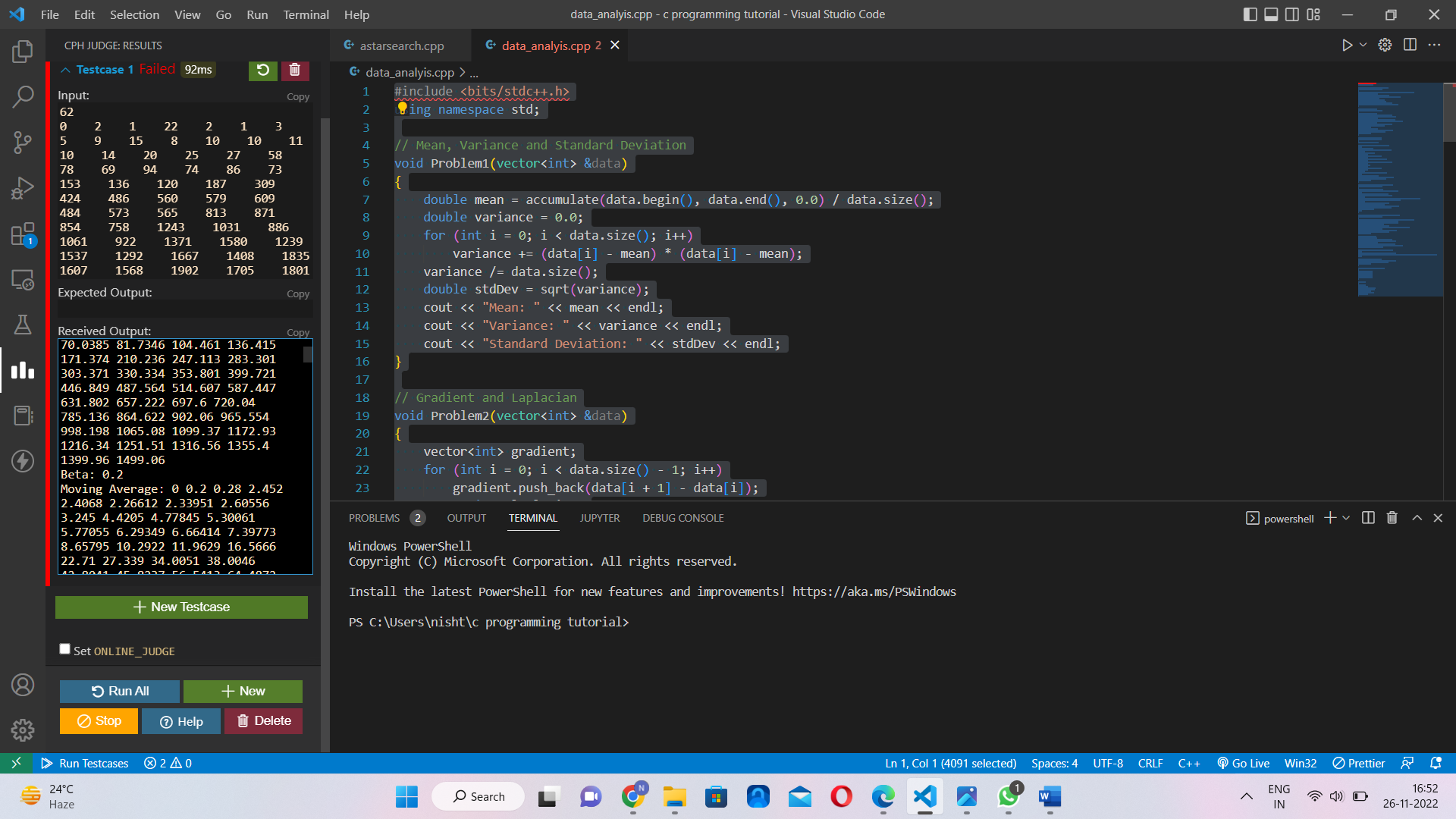


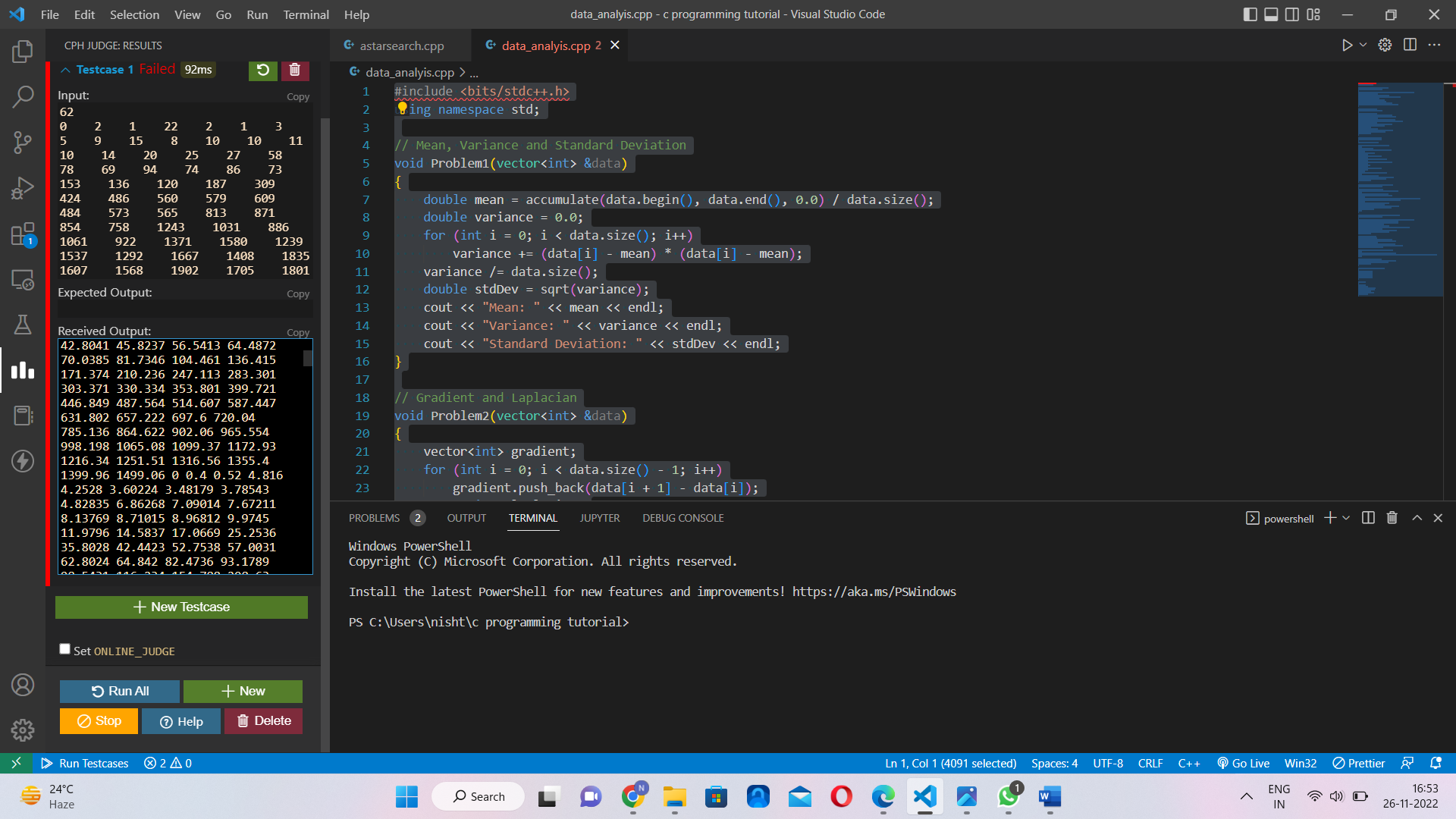
**Output:**



A screenshot of a computer

Description automatically generated





A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated