Given:

Table of Tesla Stock Closings during five consecutive trading sessions
(01 = Monday = 1/9/2023, 02 = Tuesday = 1/10/2023, …)

t	y
05	122.44
04	123.45
03	123.22
02	118.85
01	119.77

Problem Statement 5: Interpolate data into a polynomial function $P_4(t)$. Compute $P_4(t = 6)$ and explain its significance with regards to prediction.

Algorithm Description: The program first sets up the equation of form Ax = b, where A is the matrix using the inputs t, x is the coefficients we seek to find, and b is the output y vector. So, to solve the program then find the inverse matrix by finding the adjoint matrix and scaling via determinant ($A^{-1} = A_{adjoin} / \det(A)$) and simply multiplying the inverse matrix with the output vector to get the coefficient vector, $x = A^{-1}b$. The program mimics the operations the same way matrix vector multiplication, inverse is found by hand.

Pseudocode:

```
Coefficients[]
Inverse[][]
Outputs[]
Matrix[][]

Function main()
     findCoefficients()
     Output() // This is the results below, modified per image
End

Function findCoefficients()
     ConfigureMatrixA()
     FindAdjointMatrix()
     FindInverseMatrix()
     Coefficients[] = matrixVectorMultiplication(Inverse, Outputs)
End
```

```
Function ConfigureMatrixA(); //Set it up so first row and last column
is 1s just like in the lecture notes, not shown to not lengthen
pseudocode

Function FindAdjointMatrix(); //Set it up so first row and last
column is 1s just like in the lecture notes, not shown to not
lengthen pseudocode

Function FindInverseMatrix(); //Set it up so first row and last
column is 1s just like in the lecture notes, not shown to not
lengthen pseudocode

Function matrixVectorMultiplication(Inverse, Outputs)
     Vector[]

     For i in range(numRowsIn(Inverse)):
          Vector[i] = 0

          For j in range(numColsIn(Inverse)):
               Vector[i] += (Inverse[i][j] * Outputs[j])

     Return Vector
End
```

Results: (p15.java)

Points interpolated

```
Point 1: (1, 119.77000000000001)
Point 2: (2, 118.85000000000028)
Point 3: (3, 123.22000000000097)
Point 4: (4, 123.45000000000249)
Point 5: (5, 122.44000000000489)
```

Prediction + Program Runtime for Polynomial Interpolation

```
Prediction: (6, 135.4200000000087)
Operation time took 2 Milliseconds.
There were 14651 operations.
The estimated operations per second rate is 7325500
```

Interpolated Polynomial

```
0.5137500000000035x^4 + -6.709166666666661x^3 + 30.05625000000012x^2 + -51.830833333333516x^1 + 147.74000000000007
```

Observations & Analysis

The interpolated polynomial estimated by *p15.java* is approximately $P_4(t) = 0.51375x^4 - 6.709167x^3 + 30.05625x^2 - 51.830833x + 147.740$, which results in estimation of well within 4 decimal points of the data points for their given inputs. As for the program efficiency, I would say it is sufficient as it only took no more than 2-3 milliseconds to conduct about 14,650 operations. The algorithm is good; this results in a rate of 7,325,500 operations per second which does not take into account how much more time it takes to use built in math functions. I would like to note that there are more efficient ways to write the algorithm in terms of matrix vector multiplications done as well as using structures to store some values that were calculated repetitively, but because of how I have calculated program runtime, it would be meaningless for smaller inputs. The program is scalable to larger inputs for data points as the function calculating determinant of a matrix of inputs, inverse matrix finder, transposing matrix, and matrix vector multiplication are scalable to larger inputs using recursive function calls. The program is portable where it only requires a java configured environment to run the java file. Lastly, I accidentally calculated a transpose inverse matrix which I transposed to get the desired inverse matrix resulting in extra operations on top of print statements taking some of program execution time, so the actual rate of operations estimated is higher than the given rate of about 7.325 million.
