**1. Find Maximum and Minimum in an Array**

* **Problem Statement**: Write a program to find the maximum and minimum values in a single-dimensional array of integers. Use:
  + A const variable for the array size.
  + A static variable to keep track of the maximum difference between the maximum and minimum values.
  + if statements within a for loop to determine the maximum and minimum values.

A screen shot of a computer program

Description automatically generated

**2. Array Element Categorization**

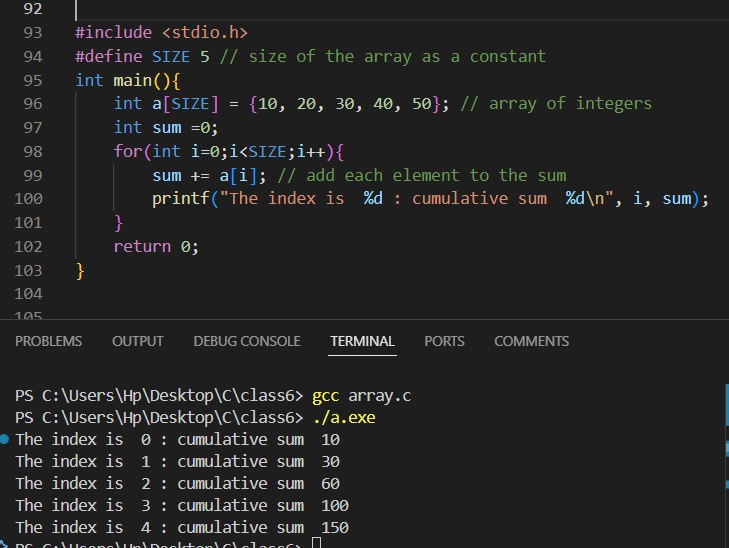
* **Problem Statement**: Categorize elements of a single-dimensional array into positive, negative, and zero values. Use:
  + A const variable to define the size of the array.
  + A for loop for traversal.
  + if-else statements to classify each element into separate arrays using static storage.

A screenshot of a computer program

Description automatically generated

**3. Cumulative Sum of Array Elements**

* **Problem Statement**: Calculate the cumulative sum of elements in a single-dimensional array. Use:
  + A static variable to hold the running total.
  + A for loop to iterate through the array and update the cumulative sum.
  + A const variable to set the array size.



**4. Check Prime Numbers in an Array**

* **Problem Statement**: Identify which elements in a single-dimensional array are prime numbers. Use:
  + A for loop to iterate through the array and check each element.
  + A nested for loop to determine if a number is prime.
  + if statements for decision-making.
  + A const variable to define the size of the array.

A screen shot of a computer program

Description automatically generated

**5. Array Rotation by N Positions**

* **Problem Statement**: Rotate the elements of a single-dimensional array to the left by N positions. Use:
  + A const variable for the rotation count.
  + A static array to store the rotated values.
  + A while loop for performing the rotation.

A screen shot of a computer program

Description automatically generated

**6. Count Frequency of Each Element**

* **Problem Statement**: Count the frequency of each unique element in a single-dimensional array. Use:
  + A const variable for the size of the array.
  + A nested for loop to compare each element with the rest.
  + A static array to store the frequency count.

A screen shot of a computer program

Description automatically generated

**7. Sort Array in Descending Order**

* **Problem Statement**: Sort a single-dimensional array in descending order using bubble sort. Use:
  + A const variable for the size of the array.
  + A nested for loop for sorting.
  + if statements for comparing and swapping elements.

A screen shot of a computer program

Description automatically generated

**8. Find the Second Largest Element**

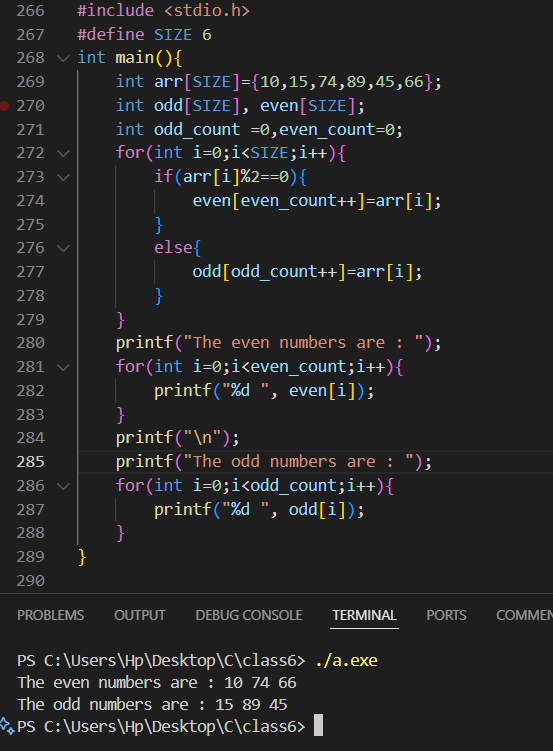
* **Problem Statement**: Find the second largest element in a single-dimensional array. Use:
  + A const variable for the array size.
  + A static variable to store the second largest element.
  + if statements and a single for loop to compare elements.

A computer screen shot of a program

Description automatically generated

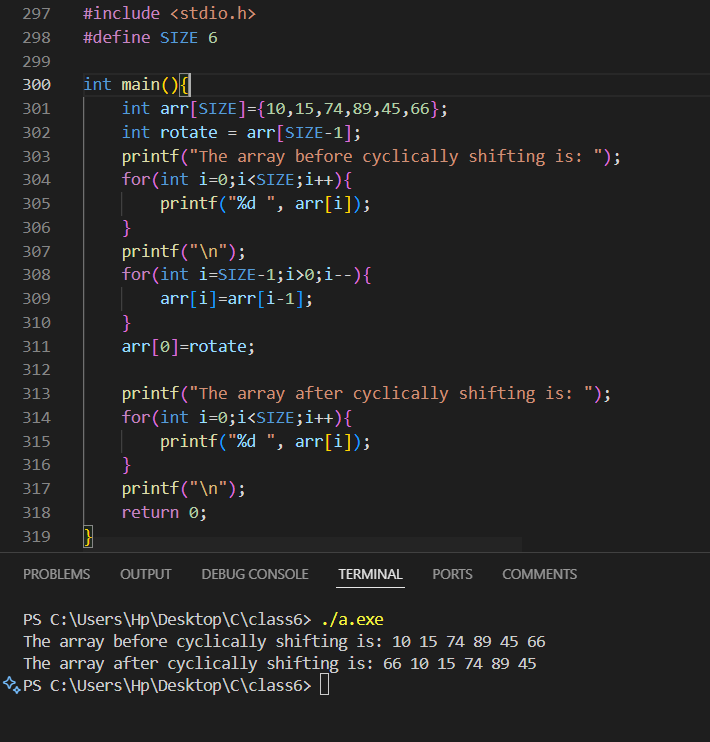
**9. Odd and Even Number Separation**

* **Problem Statement**: Separate the odd and even numbers from a single-dimensional array into two separate arrays. Use:
  + A const variable for the size of the array.
  + if-else statements to classify elements.
  + A for loop for traversal and separation.



**10. Cyclically Shift Array Elements**

* **Problem Statement**: Shift all elements of a single-dimensional array cyclically to the right by one position. Use:
  + A const variable for the array size.
  + A static variable to temporarily store the last element during shifting.
  + A for loop for the shifting operation.



**1. Engine Temperature Monitoring System**

Write a program to monitor engine temperatures at 10 different time intervals in degrees Celsius. Use:

* Proper variable declarations with const to ensure fixed limits like maximum temperature.
* Storage classes (static for counters and extern for shared variables).
* Decision-making statements to alert if the temperature exceeds a safe threshold.
* A loop to take 10 temperature readings into a single-dimensional array and check each value.

A screen shot of a computer program

Description automatically generated

**2. Fuel Efficiency Calculator**

Develop a program that calculates and displays fuel efficiency based on distances covered in 10 different trips.

* Use an array to store distances.
* Implement a loop to take inputs and calculate efficiency for each trip using a predefined fuel consumption value.
* Use volatile for sensor data inputs and conditionals to check for low efficiency (< 10 km/L).

A computer screen shot of a program

Description automatically generated

**3. Altitude Monitoring for Aircraft**

Create a program to store altitude readings (in meters) from a sensor over 10 seconds.

* Use a register variable for fast access to the current altitude.
* Store the readings in a single-dimensional array.
* Implement logic to identify if the altitude deviates by more than ±50 meters between consecutive readings.

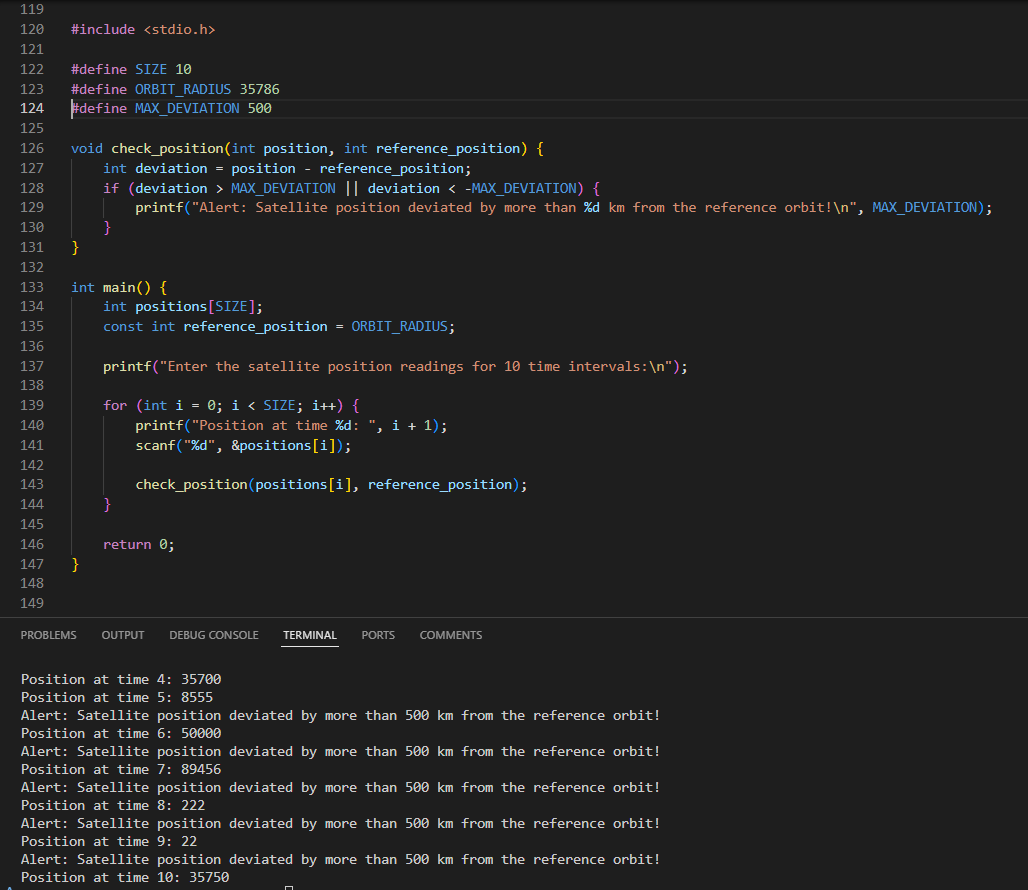
A screen shot of a computer program

Description automatically generated

**4. Satellite Orbit Analyzer**

Design a program to analyze the position of a satellite based on 10 periodic readings.

* Use const for defining the orbit radius and limits.
* Store position data in an array and calculate deviations using loops.
* Alert the user with a decision-making statement if deviations exceed specified bounds.



**5. Heart Rate Monitor**

Write a program to record and analyze heart rates from a patient during 10 sessions.

* Use an array to store the heart rates.
* Include static variables to count abnormal readings (below 60 or above 100 BPM).
* Loop through the array to calculate average heart rate and display results.

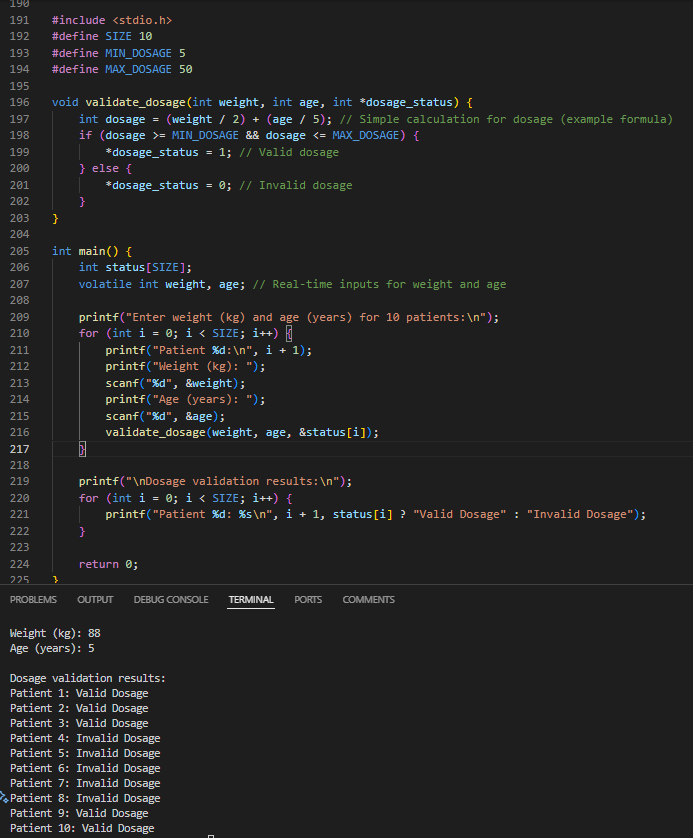
A screen shot of a computer program

Description automatically generated

**6. Medicine Dosage Validator**

Create a program to validate medicine dosage for 10 patients based on weight and age.

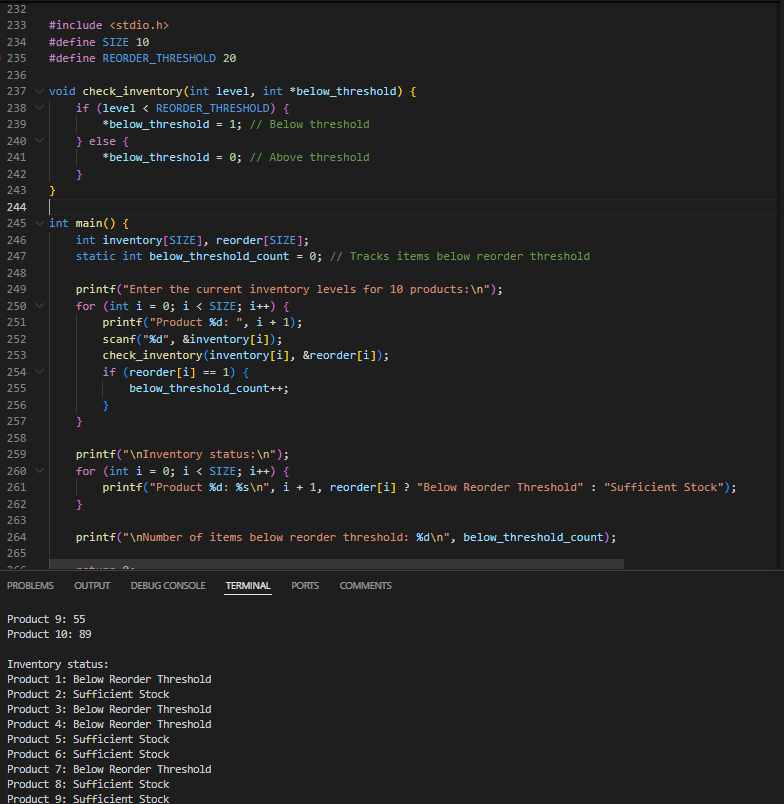
* Use decision-making statements to determine if the dosage is within safe limits.
* Use volatile for real-time input of weight and age, and store results in an array.
* Loop through the array to display valid/invalid statuses for each patient.



**7. Warehouse Inventory Tracker**

Develop a program to manage the inventory levels of 10 products.

* Store inventory levels in an array.
* Use a loop to update levels and a static variable to track items below reorder threshold.
* Use decision-making statements to suggest reorder actions.



**8. Missile Launch Codes Validator**

Develop a program to validate 10 missile launch codes.

* Use an array to store the codes.
* Use const for defining valid code lengths and formats.
* Implement decision-making statements to mark invalid codes and count them using a static variable.

A screen shot of a computer program

Description automatically generated

**9. Target Tracking System**

Write a program to track 10 target positions (x-coordinates) and categorize them as friendly or hostile.

* Use an array to store positions.
* Use a loop to process each position and conditionals to classify targets based on predefined criteria (e.g., distance from the base).
* Use register for frequently accessed decision thresholds.

A screen shot of a computer

Description automatically generated

**Problem Statements on 2 Dimensional Arrays**

**1. Matrix Addition**

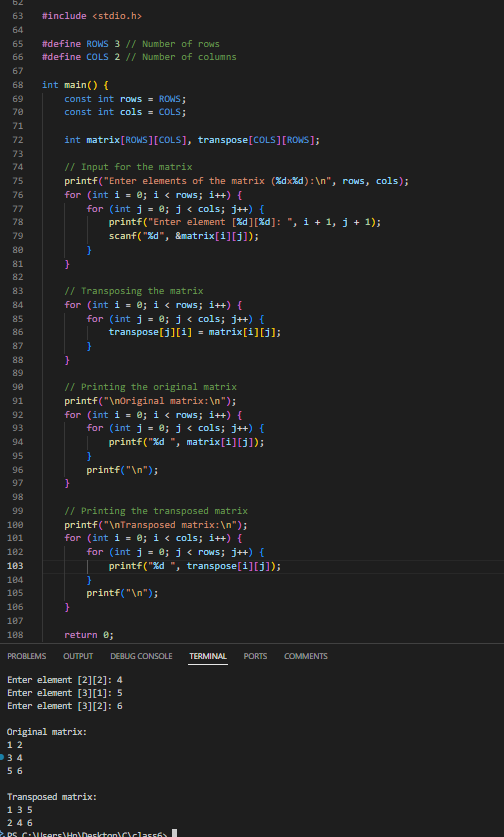
* **Problem Statement**: Write a program to perform the addition of two matrices. The program should:
  + Take two matrices as input, each of size M x N, where M and N are defined using const variables.
  + Use a static two-dimensional array to store the resulting matrix.
  + Use nested for loops to perform element-wise addition.
  + Use if statements to validate that the matrices have the same dimensions before proceeding with the addition.
* **Requirements**:
  + Declare matrix dimensions as const variables.
  + Use decision-making constructs to handle invalid dimensions.
  + Print the resulting matrix after addition.

A screen shot of a computer program

Description automatically generated

**2. Transpose of a Matrix**

* **Problem Statement**: Write a program to compute the transpose of a matrix. The program should:
  + Take a matrix of size M x N as input, where M and N are declared as const variables.
  + Use a static two-dimensional array to store the transposed matrix.
  + Use nested for loops to swap rows and columns.
  + Validate the matrix size using if statements before transposing.
* **Requirements**:
  + Print the original and transposed matrices.
  + Use a type qualifier (const) to ensure the matrix size is not modified during execution.



**3. Find the Maximum Element in Each Row**

* **Problem Statement**: Write a program to find the maximum element in each row of a two-dimensional array. The program should:
  + Take a matrix of size M x N as input, with dimensions defined using const variables.
  + Use a static array to store the maximum value of each row.
  + Use nested for loops to traverse each row and find the maximum element.
  + Use if statements to compare and update the maximum value.
* **Requirements**:
  + Print the maximum value of each row after processing the matrix.
  + Handle edge cases where rows might be empty using decision-making statements.

A screen shot of a computer program

Description automatically generated

**4. Matrix Multiplication**

* **Problem Statement**: Write a program to multiply two matrices. The program should:
  + Take two matrices as input:
    - Matrix A of size M x N
    - Matrix B of size N x P
  + Use const variables to define the dimensions M, N, and P.
  + Use nested for loops to calculate the product of the matrices.
  + Use a static two-dimensional array to store the resulting matrix.
  + Use if statements to validate that the matrices can be multiplied (N in Matrix A must equal M in Matrix B).
* **Requirements**:
  + Print both input matrices and the resulting matrix.
  + Handle cases where multiplication is invalid using decision-making constructs.

A screen shot of a computer program

Description automatically generated

**5. Count Zeros in a Sparse Matrix**

* **Problem Statement**: Write a program to determine if a given matrix is sparse. A matrix is sparse if most of its elements are zero. The program should:
  + Take a matrix of size M x N as input, with dimensions defined using const variables.
  + Use nested for loops to count the number of zero elements.
  + Use if statements to compare the count of zeros with the total number of elements.
  + Use a static variable to store the count of zeros.
* **Requirements**:
  + Print whether the matrix is sparse or not.
  + Use decision-making statements to handle matrices with no zero elements.
  + Validate matrix dimensions before processing.

A screen shot of a computer program

Description automatically generated

**Problem Statements on 3 Dimensional Arrays**

**1. 3D Matrix Addition**

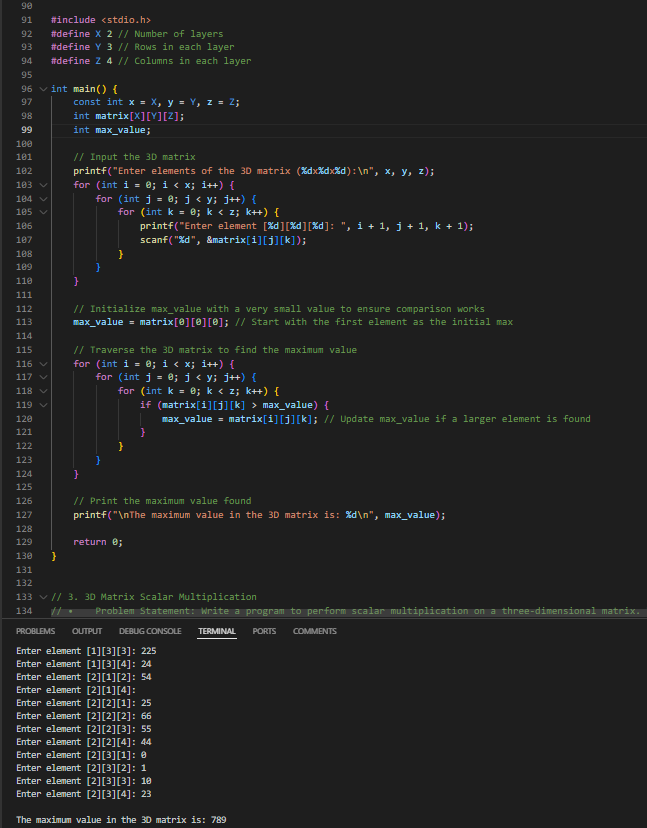
* **Problem Statement**: Write a program to perform element-wise addition of two three-dimensional matrices. The program should:
  + Take two matrices as input, each of size X x Y x Z, where X, Y, and Z are defined using const variables.
  + Use a static three-dimensional array to store the resulting matrix.
  + Use nested for loops to iterate through the elements of the matrices.
  + Use if statements to validate that the dimensions of both matrices are the same before performing addition.
* **Requirements**:
  + Declare matrix dimensions as const variables.
  + Use decision-making statements to handle mismatched dimensions.
  + Print the resulting matrix after addition.

A screenshot of a computer program

Description automatically generated

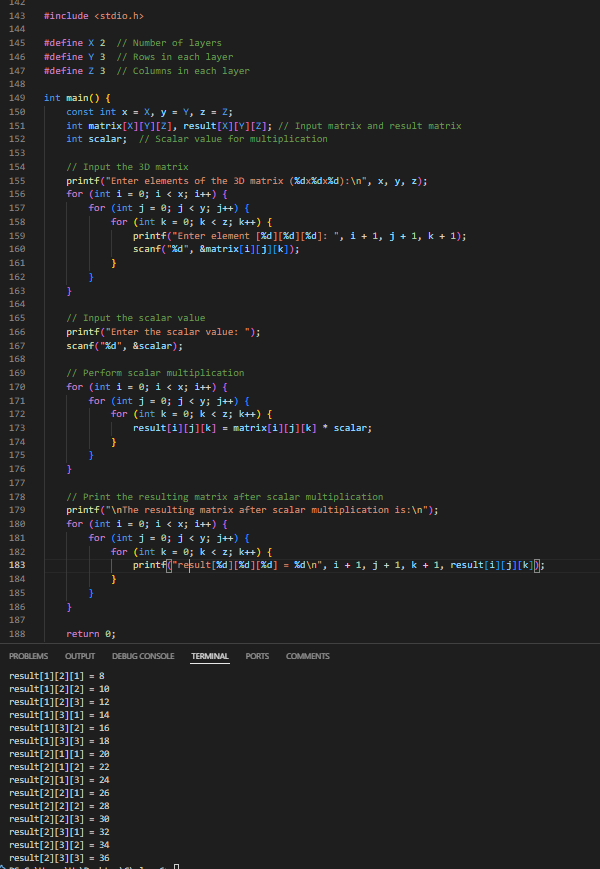
**2. Find the Maximum Element in a 3D Array**

* **Problem Statement**: Write a program to find the maximum element in a three-dimensional matrix. The program should:
  + Take a matrix of size X x Y x Z as input, where X, Y, and Z are declared as const variables.
  + Use a static variable to store the maximum value found.
  + Use nested for loops to traverse all elements of the matrix.
  + Use if statements to compare and update the maximum value.
* **Requirements**:
  + Print the maximum value found in the matrix.
  + Handle edge cases where the matrix might contain all negative numbers or zeros using decision-making statements.



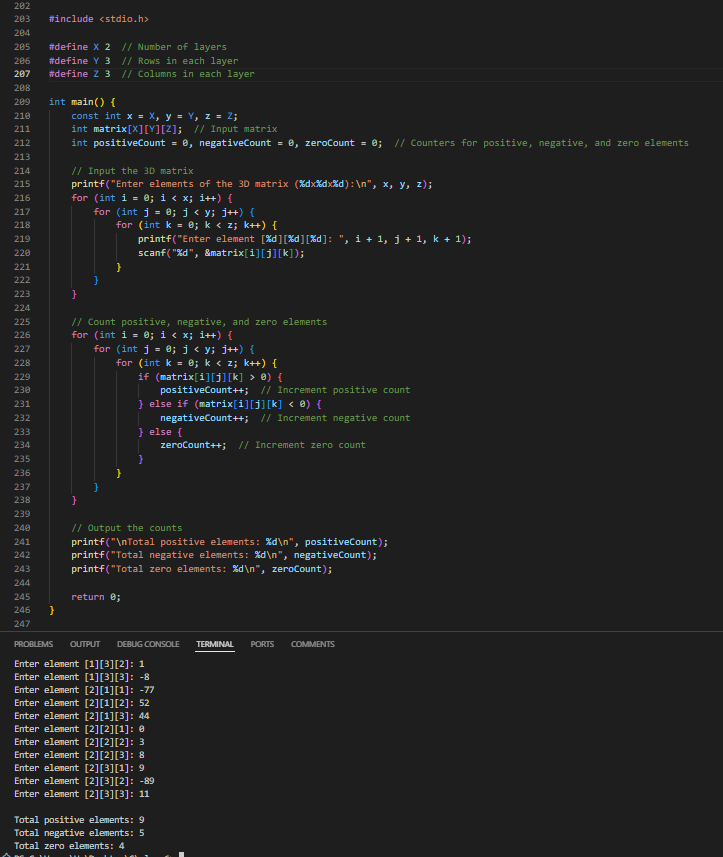
**3. 3D Matrix Scalar Multiplication**

* **Problem Statement**: Write a program to perform scalar multiplication on a three-dimensional matrix. The program should:
  + Take a matrix of size X x Y x Z and a scalar value as input, where X, Y, and Z are declared as const variables.
  + Use a static three-dimensional array to store the resulting matrix.
  + Use nested for loops to multiply each element of the matrix by the scalar.
* **Requirements**:
  + Print the original matrix and the resulting matrix after scalar multiplication.
  + Use decision-making statements to handle invalid scalar values (e.g., zero or negative scalars) if necessary.



**4. Count Positive, Negative, and Zero Elements in a 3D Array**

* **Problem Statement**: Write a program to count the number of positive, negative, and zero elements in a three-dimensional matrix. The program should:
  + Take a matrix of size X x Y x Z as input, where X, Y, and Z are defined using const variables.
  + Use three static variables to store the counts of positive, negative, and zero elements, respectively.
  + Use nested for loops to traverse the matrix.
  + Use if-else statements to classify each element.
* **Requirements**:
  + Print the counts of positive, negative, and zero elements.
  + Ensure edge cases (e.g., all zeros or all negatives) are handled correctly.



**5. Transpose of a 3D Matrix Along a Specific Axis**

* **Problem Statement**: Write a program to compute the transpose of a three-dimensional matrix along a specific axis (e.g., swap rows and columns for a specific depth). The program should:
  + Take a matrix of size X x Y x Z as input, where X, Y, and Z are defined using const variables.
  + Use a static three-dimensional array to store the transposed matrix.
  + Use nested for loops to perform the transpose operation along the specified axis.
  + Use if statements to validate the chosen axis for transposition.
* **Requirements**:
  + Print the original matrix and the transposed matrix.
  + Ensure invalid axis values are handled using decision-making constructs.

