Trainer: Asif Nafees

1. Inputting Elements into an Array:
Use a loop to input elements from the user.

```
#include <stdio.h>
int main() {
   int arr[5];
   printf("Enter 5 elements:\n");
   for (int i = 0; i < 5; i++) {
      scanf("%d", &arr[i]); // Taking input from the user
   }
   return 0;
}
```

2. Outputting Elements of an Array:
Use a loop to display elements.

```
#include <stdio.h>
int main() {
   int arr[5] = {1, 2, 3, 4, 5};
   printf("Array elements are:\n");
   for (int i = 0; i < 5; i++) {
      printf("%d ", arr[i]); // Printing each element
   }
   return 0;
}
```

Basic Problems in 1D Array

1. Problem 1: Find the Sum of All Elements in an Array

```
#include <stdio.h>
int main() {
   int n, sum = 0;
   printf("Enter the number of elements: ");
   scanf("%d", &n);
   int arr[n];
   printf("Enter %d elements:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
      sum += arr[i]; // Adding each element to sum
   }
   printf("Sum of all elements: %d\n", sum);
   return 0;
```

```
}
```

2. Problem 2: Find the Largest Element in an Array

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int largest = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > largest) {
            largest = arr[i];
        }
    }
    printf("Largest element: %d\n", largest);
    return 0;
}
```

3. Problem 3: Reverse the Elements of an Array

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Reversed array: ");
    for (int i = n - 1; i >= 0; i--) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

Multi-Dimensional Arrays in C Language:
A multi-dimensional array is an array of arrays. The most commonly used multi-dimensional array is the two-dimensional array. A two-dimensional array is structured as a table of rows and columns. The elements of a two-dimensional array are accessed using two subscripts: one for the row and one for the column.

Declaring and Initializing a Two-Dimensional Array

Syntax for Declaring a 2D Array:
data_type array_name[rows][columns];

data_type: Type of elements to be stored (e.g., int, float, char).
array_name: Name of the array.
rows: Number of rows in the array.
columns: Number of columns in the array.

Example:
int matrix[3][4]; // Declares a 2D array named 'matrix' with 3 rows and 4 columns.

Initialization of 2D Arrays:
1. Method 1: Initializing All Elements
int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}}; // A 2x3 matrix
Here, matrix[0][0] = 1, matrix[0][1] = 2, ..., matrix[1][2] = 6.

2. Method 2: Partial Initialization
int matrix[2][3] = {{1, 2}, {4}}; // Uninitialized elements are set to 0
This initializes the matrix as:
1 2 0
4 0 0

3. Accessing Elements in a 2D Array:
Use nested loops to access elements of a 2D array.
Example:
printf("%d", matrix[0][1]); // Accesses and prints the element in the first row, second column.

1. Taking Input for a 2D Array:

```c
#include <stdio.h>
int main() {
    int matrix[3][3];
    printf("Enter elements for a 3x3 matrix:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            scanf("%d", &matrix[i][j]); // Taking input for each element
        }
    }
    return 0;
}
```

2. Displaying a 2D Array:

```c
#include <stdio.h>
int main() {
    int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    printf("The 3x3 matrix is:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matrix[i][j]); // Printing each element
        }
        printf("\n"); // Move to the next line after each row
    }
    return 0;
}
```

Basic Problems with 2D Arrays (Matrix Problems)

1. Problem 1: Sum of All Elements in a Matrix

```c
#include <stdio.h>
int main() {
    int rows, columns;
    printf("Enter the number of rows and columns: ");
    scanf("%d %d", &rows, &columns);

    int matrix[rows][columns];
    printf("Enter elements of the matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            scanf("%d", &matrix[i][j]);
        }
```

```
    }

    int sum = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            sum += matrix[i][j]; // Adding each element to the sum
        }
    }

    printf("Sum of all elements: %d\n", sum);
    return 0;
}
```

2. Problem 2: Transpose of a Matrix

The transpose of a matrix is obtained by interchanging its rows and columns.

```
#include <stdio.h>
int main() {
    int rows, columns;
    printf("Enter the number of rows and columns: ");
    scanf("%d %d", &rows, &columns);

    int matrix[rows][columns];
    printf("Enter elements of the matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
    printf("Transpose of the matrix:\n");
    for (int i = 0; i < columns; i++) {
        for (int j = 0; j < rows; j++) {
            printf("%d ", matrix[j][i]); // Transposing elements
        }
        printf("\n");
    }
    return 0;
}
```

3. Problem 3: Matrix Addition

To add two matrices, both must have the same dimensions.

```
#include <stdio.h>
int main() {
```

```
    int rows, columns;
    printf("Enter the number of rows and columns: ");
    scanf("%d %d", &rows, &columns);
    int matrix1[rows][columns], matrix2[rows][columns], sum[rows][columns];
    printf("Enter elements of the first matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            scanf("%d", &matrix1[i][j]);
        }
    }
    printf("Enter elements of the second matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            scanf("%d", &matrix2[i][j]);
        }
    }
    // Adding corresponding elements
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            sum[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
    printf("Sum of the two matrices:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            printf("%d ", sum[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

4. Problem 4: Matrix Multiplication

To multiply two matrices, the number of columns in the first matrix must be equal to the number of rows in the second matrix.

```
#include <stdio.h>
int main() {
    int rows1, columns1, rows2, columns2;
    printf("Enter rows and columns of the first matrix: ");
    scanf("%d %d", &rows1, &columns1);
    printf("Enter rows and columns of the second matrix: ");
    scanf("%d %d", &rows2, &columns2);

    if (columns1 != rows2) {
```

```
        printf("Matrix multiplication not possible.\n");
        return 0;
    }
    int matrix1[rows1][columns1], matrix2[rows2][columns2], product[rows1][columns2];
    printf("Enter elements of the first matrix:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < columns1; j++) {
            scanf("%d", &matrix1[i][j]);
        }
    }
    printf("Enter elements of the second matrix:\n");
    for (int i = 0; i < rows2; i++) {
        for (int j = 0; j < columns2; j++) {
            scanf("%d", &matrix2[i][j]);
        }
    }
    // Initializing the product matrix to 0
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < columns2; j++) {
            product[i][j] = 0;
        }
    }
    // Multiplying matrices
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < columns2; j++) {
            for (int k = 0; k < columns1; k++) {
                product[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
    printf("Product of the two matrices:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < columns2; j++) {
            printf("%d ", product[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```