1. Matrix Addition

```
#include <stdio.h>

int main() {
    int m, n;
    printf("Enter the size of the matrix (rows and columns): ");
    scanf("%d %d", &m, &n);

    int A[m][n], B[m][n], sum[m][n];

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

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

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            sum[i][j] = A[i][j] + B[i][j];
        }
    }

    printf("Resultant matrix:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", sum[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

2. Matrix Multiplication

```
#include <stdio.h>

int main() {
    int r1, c1, r2, c2;

    printf("Enter the size of the first matrix (rows and columns): ");
    scanf("%d %d", &r1, &c1);
    printf("Enter the size of the second matrix (rows and columns): ");
    scanf("%d %d", &r2, &c2);

    if (c1 != r2) {
        printf("Matrix multiplication not possible.\n");
        return 1;
    }
    int A[r1][c1], B[r2][c2], result[r1][c2];
    printf("Enter elements of first matrix:\n");
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c1; j++) {
            scanf("%d", &A[i][j]);
        }
    }
    printf("Enter elements of second matrix:\n");
    for (int i = 0; i < r2; i++) {
        for (int j = 0; j < c2; j++) {
            scanf("%d", &B[i][j]);
        }
    }
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            result[i][j] = 0;
            for (int k = 0; k < c1; k++) {
                result[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    printf("Resultant matrix:\n");
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

3. Transpose of a Matrix

```
#include <stdio.h>

int main() {
    int m, n;
    printf("Enter the size of the matrix (rows and columns): ");
    scanf("%d %d", &m, &n);

    int matrix[m][n], transpose[n][m];

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

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            transpose[j][i] = matrix[i][j];
        }
    }

    printf("Transposed matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            printf("%d ", transpose[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

4. Find the Largest Element

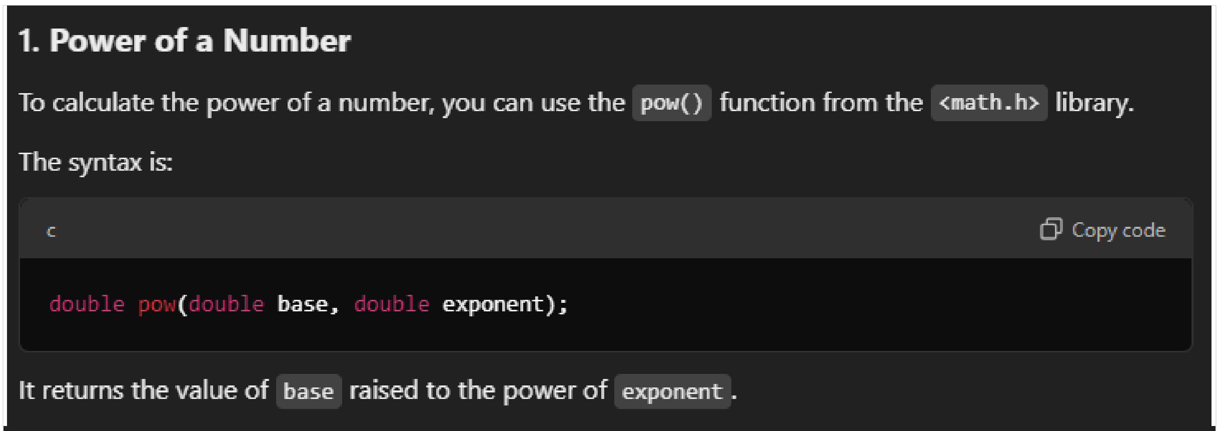
```
#include <stdio.h>

int main() {
    int m, n;
    printf("Enter the size of the matrix (rows and columns): ");
    scanf("%d %d", &m, &n);

    int matrix[m][n];

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

    int max = matrix[0][0];
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (matrix[i][j] > max) {
                max = matrix[i][j];
            }
        }
    }

    printf("Largest element: %d\n", max);

    return 0;
}
```

5. Sum of Diagonal Elements

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the size of the square matrix (n x n): ");
    scanf("%d", &n);

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

    int principalSum = 0, secondarySum = 0;
    for (int i = 0; i < n; i++) {
        principalSum += matrix[i][i];
        secondarySum += matrix[i][n - 1 - i];
    }

    printf("Sum of principal diagonal: %d\n", principalSum);
    printf("Sum of secondary diagonal: %d\n", secondarySum);

    return 0;
}
```

Maximum in Each Row and Column:

```c
#include <stdio.h>

int main() {
    int a, b;
    printf("Enter the number of rows and columns: ");
    scanf("%d %d", &a, &b);

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

    // Find maximum of each row
    printf("Maximum of each row:\n");
    for (int i = 0; i < a; i++) {
        int rowMax = mat[i][0];
        for (int j = 1; j < b; j++) {
            if (mat[i][j] > rowMax) {
                rowMax = mat[i][j];
            }
        }
        printf("Row %d: %d\n", i + 1, rowMax);
    }

    // Find maximum of each column
    printf("Maximum of each column:\n");
    for (int j = 0; j < b; j++) {
        int colMax = mat[0][j];
        for (int i = 1; i < a; i++) {
            if (mat[i][j] > colMax) {
                colMax = mat[i][j];
            }
        }
        printf("Column %d: %d\n", j + 1, colMax);
    }

    return 0;
}
```



1. Power of a Number

To calculate the power of a number, you can use the `pow()` function from the `<math.h>` library.

The syntax is:

```c
double pow(double base, double exponent);
```

It returns the value of `base` raised to the power of `exponent`.

Example: Power of a Number

```
#include <stdio.h>
#include <math.h>  // Required for pow()

int main() {
    double base, exponent, result;
    printf("Enter base and exponent: ");
    scanf("%lf %lf", &base, &exponent);  // Read base and exponent

    result = pow(base, exponent);  // Calculate base raised to exponent
    printf("%.2f raised to %.2f is: %.2f\n", base, exponent, result);

    return 0;
}
```

2. Root of a Number

```
#include <stdio.h>
#include <math.h>  // Required for pow()

int main() {
    double number, n, result;
    printf("Enter the number and the root (n): ");
    scanf("%lf %lf", &number, &n);  // Read number and root

    result = pow(number, 1.0 / n);  // Calculate nth root
    printf("The %.2fth root of %.2f is: %.2f\n", n, number, result);

    return 0;
}
```