Practical Work Series

Exercise 01: Sum of Elements
Write a program that calculates and prints the sum of all elements in an integer array.

Exercise 2: Finding the MAX
Element Write a program that finds and prints the MAX element in an array.

Exercise 3: Reverse an Array
Write a program that reverses the elements of an array.

Exercise 4: Matrix Multiplication
Write a program that multiplies two matrices and prints the result. You can define matrices as 2D arrays and assume that the matrices are compatible for multiplication (the number of columns in the first matrix is equal to the number of rows in the second matrix).

Exercise 5: Concatenate Strings Dynamically
Write a program that concatenates two strings dynamically. You need to allocate memory for the resulting string based on the lengths of the input strings.

Exercise 6: Dynamic 2D Array Operations
Write a program that dynamically creates a 2D array, initializes it, and performs operations like finding the sum of elements, transposing the matrix, and freeing the allocated memory.

Solutions

Exercise 01

```
#include <stdio.h>

int main() {
   int arr[] = {1, 2, 3, 4, 5};
   int size = sizeof(arr) / sizeof(arr[0]);

   int sum = 0;
   for (int i = 0; i < size; i++) {
      sum += arr[i];
```

```
    }

    printf("Sum of elements: %d\n", sum);

    return 0;
}
```

Exercise 02

```
#include <stdio.h>

int main() {
    int arr[] = {5, 2, 8, 10, 3};
    int size = sizeof(arr) / sizeof(arr[0]);

    int max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }

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

    return 0;
}
```

Exercise 03

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    for (int i = 0; i < size / 2; i++) {
        // Swap elements at i and size - 1 - i
        int temp = arr[i];
        arr[i] = arr[size - 1 - i];
```

```
        arr[size - 1 - i] = temp;
    }

    printf("Reversed array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Exercise 04

```
#include <stdio.h>

#define ROWS_A 2
#define COLS_A 3
#define ROWS_B 3
#define COLS_B 2

void multiplyMatrices(int A[ROWS_A][COLS_A], int B[ROWS_B][COLS_B], int
result[ROWS_A][COLS_B]) {
    for (int i = 0; i < ROWS_A; i++) {
        for (int j = 0; j < COLS_B; j++) {
            result[i][j] = 0;
            for (int k = 0; k < COLS_A; k++) {
                result[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

void printMatrix(int matrix[ROWS_A][COLS_B]) {
    for (int i = 0; i < ROWS_A; i++) {
        for (int j = 0; j < COLS_B; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
```

```
}

int main() {
    int matrixA[ROWS_A][COLS_A] = {{1, 2, 3}, {4, 5, 6}};
    int matrixB[ROWS_B][COLS_B] = {{7, 8}, {9, 10}, {11, 12}};
    int resultMatrix[ROWS_A][COLS_B];

    multiplyMatrices(matrixA, matrixB, resultMatrix);

    printf("Matrix A:\n");
    printMatrix(matrixA);

    printf("\nMatrix B:\n");
    printMatrix(matrixB);

    printf("\nResult Matrix:\n");
    printMatrix(resultMatrix);

    return 0;
}
```

Exercise 05

```
#include <stdio.h>

void bubbleSort(int arr[], int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            // Swap if the element found is greater than the next element
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

void printArray(int arr[], int size) {
```

```
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr[] = {64, 25, 12, 22, 11};
    int size = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    printArray(arr, size);

    bubbleSort(arr, size);

    printf("Sorted array: ");
    printArray(arr, size);

    return 0;
}

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* concatenateStrings(const char* str1, const char* str2) {
    // Calculate the length of the concatenated string
    size_t len1 = strlen(str1);
    size_t len2 = strlen(str2);
    size_t totalLength = len1 + len2 + 1; // +1 for the null terminator

    // Allocate memory for the concatenated string
    char* result = (char*)malloc(totalLength * sizeof(char));

    // Check if memory allocation was successful
    if (result == NULL) {
        perror("Memory allocation failed");
        exit(EXIT_FAILURE);
    }
```

```
    // Copy the first string into the result
    strcpy(result, str1);

    // Concatenate the second string to the result
    strcat(result, str2);

    return result;
}
```

Exercise 06

```
int main() {
    const char* string1 = "Hello, ";
    const char* string2 = "world!";

    // Concatenate strings dynamically
    char* concatenatedString = concatenateStrings(string1, string2);

    // Print the result
    printf("Concatenated string: %s\n", concatenatedString);

    // Free the dynamically allocated memory
    free(concatenatedString);

    return 0;
}
```