# Exercises - Problems Sheet # 2: One/Two Dimensional Arrays

**Spring 2024**

**No. Of Questions: 14** **No. Of Pages: 7**

## Answer the following:

1) Write a C program that calculates the maximum, the minimum, and the average of 10 numbers entered by the user and stored in an array.

```c
#include <stdio.h>

int main() {
    int numbers[10];
    int max, min, sum = 0;
    float average;

    printf("Enter 10 numbers:\n");
    for (int i = 0; i < 10; i++) {
        printf("%d: ", i + 1);
        scanf("%d", &numbers[i]);
        sum += numbers[i];
    }

    max = min = numbers[0];

    for (int i = 1; i < 10; i++) {
        if (numbers[i] > max) {
            max = numbers[i];
        }
        if (numbers[i] < min) {
            min = numbers[i];
        }
    }

    average = (float)sum / 10;

    printf("Maximum: %d\n", max);
    printf("Minimum: %d\n", min);
    printf("Average: %.2f\n", average);

    return 0;
}
```

2) Write a C function that checks if an array is in an ascending order.

```c
#include <stdio.h>
#include <stdbool.h>

bool isArrayInAscendingOrder(int arr[], int size);

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

    if (isArrayInAscendingOrder(arr, size)) {
        printf("The array is in ascending order.\n");
    } else {
        printf("The array is not in ascending order.\n");
    }

    return 0;
}

bool isArrayInAscendingOrder(int arr[], int size) {
    for (int i = 0; i < size - 1; i++) {
        if (arr[i] > arr[i + 1]) {
            return false;
        }
    }
    return true;
}
```

3) Write a C program that reads 10 integer numbers from the user, and then the program should calculate the sum of the odd numbers, and the sum of the even numbers.

```c
#include <stdio.h>

int main() {
    int numbers[10];
    int sumOdd = 0;
    int sumEven = 0;
    int i;

    printf("Enter 10 integers:\n");

    for(i = 0; i < 10; i++) {
        scanf("%d", &numbers[i]);
        if(numbers[i] % 2 == 0) {
            sumEven += numbers[i];
        } else {
            sumOdd += numbers[i];
        }
    }

    printf("Sum of even numbers: %d\n", sumEven);
    printf("Sum of odd numbers: %d\n", sumOdd);

    return 0;
}
```

4) Write a C program that asks the user to enter 10 integers in an array. he program will then display (based on the entered numbers) one of the following messages

the numbers in the array are increasing ,
the numbers in the array are decreasing ,
the numbers in the array are not changing , or
the numbers in the array are increasing and then decreasing.

```c
#include <stdio.h>

int main() {
    int numbers[10];
    int i;
    int increasing = 1, decreasing = 1, notChanging = 1;

    printf("Please enter 10 integers:\n");
    for(i = 0; i < 10; i++) {
        scanf("%d", &numbers[i]);
    }

    for(i = 0; i < 9; i++) {
        if(numbers[i] < numbers[i + 1]) {
            decreasing = 0;
            notChanging = 0;
        }
        else if(numbers[i] > numbers[i + 1]) {
            increasing = 0;
            notChanging = 0;
        }
        else {
            increasing = 0;
            decreasing = 0;
        }
    }

    if(notChanging)
        printf("the numbers in the array are not changing\n");
    else if(increasing)
        printf("the numbers in the array are increasing\n");
    else if(decreasing)
        printf("the numbers in the array are decreasing\n");
    else
        printf("the numbers in the array are increasing and then decreasing\n");

    return 0;
}
```

5) Write a C program that reads a matrix ( □ ), and asks the user to choose a number, and then displays the position of the selected number if found, otherwise it displays number not found.

```c
#include <stdio.h>

int main() {
    int numbers[10];
    int i;
    int increasing = 1, decreasing = 1, notChanging = 1;

    printf("Please enter 10 integers:\n");
    for(i = 0; i < 10; i++) {
        scanf("%d", &numbers[i]);
    }

    for(i = 0; i < 9; i++) {
        if(numbers[i] < numbers[i + 1]) {
            decreasing = 0;
            notChanging = 0;
        }
        else if(numbers[i] > numbers[i + 1]) {
            increasing = 0;
            notChanging = 0;
        }
        else {
            increasing = 0;
            decreasing = 0;
        }
    }

    if(notChanging)
        printf("the numbers in the array are not changing\n");
    else if(increasing)
        printf("the numbers in the array are increasing\n");
    else if(decreasing)
        printf("the numbers in the array are decreasing\n");
    else
        printf("the numbers in the array are increasing and then decreasing\n");

    return 0;
}
```

6) Write a C program to read a matrix from the user, and then display the row with the maximum total/sum *(that is, the row whose sum of elements is maximum)*.

```c
#include <stdio.h>

int main() {
    int rows, cols, i, j, sum, maxSum = 0, maxRow = 0;

    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    printf("Enter the number of columns: ");
    scanf("%d", &cols);

    int matrix[rows][cols];

    printf("Enter the elements of the matrix:\n");
    for(i = 0; i < rows; i++) {
        for(j = 0; j < cols; j++) {
            printf("Enter element [%d][%d]: ", i + 1, j + 1);
            scanf("%d", &matrix[i][j]);
        }
    }

    for(i = 0; i < rows; i++) {
        sum = 0;
        for(j = 0; j < cols; j++) {
            sum += matrix[i][j];
        }
        if(sum > maxSum) {
            maxSum = sum;
            maxRow = i;
        }
    }

    printf("Row with the maximum sum is Row %d with a sum of %d\n", maxRow + 1, maxSum);

    return 0;
}
```

7) By using a two-dimensional array, write a C program to display the matrix shown below:

| | | | | |
|---|---|---|---|---|
| **0** | **1** | **1** | **1** | **1** |
| **-1** | **0** | **1** | **1** | **1** |
| **-1** | **-1** | **0** | **1** | **1** |
| **-1** | **-1** | **-1** | **0** | **1** |
| **-1** | **-1** | **-1** | **-1** | **0** |

```c
#include <stdio.h>

int main() {
    int matrix[5][5];
    int i, j;

    for (i = 0; i < 5; i++) {
        for (j = 0; j < 5; j++) {
            if (i < j) {
                matrix[i][j] = 1;
            } else if (i == j) {
                matrix[i][j] = 0;
            } else {
                matrix[i][j] = -1;
            }
        }
    }

    for (i = 0; i < 5; i++) {
        for (j = 0; j < 5; j++) {
            printf("%2d ", matrix[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

8) By using a two-dimensional array, write a C program to display a Pascal triangle of any size. In a Pascal triangle, the first & second rows are set to 1. Each element of the triangle (starting from the third row downwards) is the sum of the element directly above it and the element to the left of the element directly above it. ***See the following example of a Pascal triangle (with a size = 5):***

| | | | | |
|---|---|---|---|---|
| **1** | | | | |
| **1** | **1** | | | |
| **1** | **2** | **1** | | |
| **1** | **3** | **3** | **1** | |
| **1** | **4** | **6** | **4** | **1** |

```c
#include <stdio.h>

void printPascal(int n) {
    int arr[n][n];

    for (int line = 0; line < n; line++) {
        for (int i = 0; i <= line; i++) {
            if (line == i || i == 0)
                arr[line][i] = 1;
            else
                arr[line][i] = arr[line-1][i-1] + arr[line-1][i];
            printf("%d ", arr[line][i]);
        }
        printf("\n");
    }
}

int main() {
    int n;
    scanf("%d",&n);
    printPascal(n);
    return 0;
}
```

9) Write a C function that reverses an array in another one.

```c
#include <stdio.h>

void reverseArray(int original[], int size, int reversed[]) {
    for (int i = 0; i < size; i++) {
        reversed[size - 1 - i] = original[i];
    }
}

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

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

    reverseArray(original, size, reversed);

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

    printf("Reversed array: ");
    printArray(reversed, size);

    return 0;
}
```

10) Write a C function that reads the number of students in a class, and five grades for each student. The function should then compute the average grade for each student.

```c
#include <stdio.h>

void calculateAverageGrades(int numberOfStudents) {
    float grades[5];
    float sum, average;

    for (int i = 0; i < numberOfStudents; i++) {
        sum = 0;
        for (int j = 0; j < 5; j++) {
            scanf("%f", &grades[j]);
            sum += grades[j];
        }

        average = sum / 5;
        printf("%.2f\n", average);
    }
}

int main() {
    int numberOfStudents;

    scanf("%d", &numberOfStudents);
    calculateAverageGrades(numberOfStudents);

    return 0;
}
```

11) Write a C function that checks if a matrix is sparse or not. Given that a sparse matrix is a matrix in which most of the elements are zero *(that is, the number of zero-valued elements are more than 50% of the total number of elements)*.

```c
#include <stdio.h>

int isSparse(int rows, int cols, int matrix[rows][cols]) {
    int zeroCount = 0;
    int totalCount = rows * cols;

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix[i][j] == 0) {
                zeroCount++;
            }
        }
    }

    return zeroCount > totalCount / 2;
}

int main() {
    int matrix[4][5] = {
        {0, 0, 3, 0, 4},
        {0, 0, 0, 0, 2},
        {0, 0, 0, 0, 0},
        {0, 2, 6, 0, 0}
    };

    if (isSparse(4, 5, matrix)) {
        printf("The matrix is sparse.\n");
    } else {
        printf("The matrix is not sparse.\n");
    }

    return 0;
}
```

12) Write a C function that reads a matrix and checks whether the given matrix is a symmetric matrix or not. Given that: If a square matrix A is equal to its transpose $A^T$, then it is a symmetric matrix. For example: if the elements of the matrix are:

| | | |
|---|---|---|
| **1** | **2** | **3** |
| **2** | **4** | **5** |
| **3** | **5** | **8** |

Then the matrix is symmetric

```c
#include <stdio.h>
#define SIZE 3

void readMatrix(int matrix[SIZE][SIZE]) {
    for(int i = 0; i < SIZE; i++) {
        for(int j = 0; j < SIZE; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

int isSymmetric(int matrix[SIZE][SIZE]) {
    for(int i = 0; i < SIZE; i++) {
        for(int j = 0; j < SIZE; j++) {
            if(matrix[i][j] != matrix[j][i]) {
                return 0;
            }
        }
    }
    return 1;
}

int main() {
    int matrix[SIZE][SIZE];

    readMatrix(matrix);

    if(isSymmetric(matrix)) {
        printf("The matrix is symmetric.\n");
    } else {
        printf("The matrix is not symmetric.\n");
    }

    return 0;
}
```

13) Write a C program that reads the radius of a circle, and then calls a function that returns the circumference and the area of that circle. The program should include a global constant variable.

```c
#include <stdio.h>
#define PI 3.14159

void calculateCircle(float radius, float *circumference, float *area) {
    *circumference = 2 * PI * radius;
    *area = PI * radius * radius;
}

int main() {
    float radius, circumference, area;

    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);

    calculateCircle(radius, &circumference, &area);

    printf("Circumference of the circle: %.2f\n", circumference);
    printf("Area of the circle: %.2f\n", area);

    return 0;
}
```

14) Write a C program that reads a 1-D array of any size, then calls a function that returns the following:

- The maximum value in the array.
- The minimum value in the array.
- The average value of the array.

```c
#include <stdio.h>

void findMinMaxAvg(int arr[], int size, int *max, int *min, double *avg);

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

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

    int max, min;
    double avg;
    findMinMaxAvg(arr, n, &max, &min, &avg);

    printf("Maximum value in the array is: %d\n", max);
    printf("Minimum value in the array is: %d\n", min);
    printf("Average value of the array is: %.2f\n", avg);

    return 0;
}

void findMinMaxAvg(int arr[], int size, int *max, int *min, double *avg) {
    *max = arr[0];
    *min = arr[0];
    double sum = 0.0;

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

    *avg = sum / size;
}
```