Lab 4
Array

In this lab, we will learn about **arrays**, a fundamental data structure in C/C++ that allows us to store multiple values of the same type in a single variable.

Instructions

1. Declaring and Initializing Arrays

An array is a collection of elements, all of the same type, stored in contiguous memory locations. The syntax for declaring an array includes specifying the type, the array name, and the size.

- **Declaration**: Specifies the type and size of the array.

```
dataType arrayName[size];
```

  For example:

```
int arr[5];
```

- **Initialization**: Arrays can be initialized at the time of declaration.

  For example:

```
int arr[5]  = {1, 2, 3, 4, 5};  // Arr has 5 elements: 1, 2, 3, 4, 5
int arr[]   = {1, 2, 3, 4, 5};  // Arr has 5 elements: 1, 2, 3, 4, 5
int arr[5]  = {1, 2};           // Arr has 5 elements: 1, 2, 0, 0, 0
int arr[5]  = {0};              // Arr has 5 elements: 0, 0, 0, 0, 0
```

2. Accessing and Modifying Array Elements

Array elements can be accessed and modified using indices, starting from 0 for the first element.

- Access an element by specifying the index in square brackets.

```
int x = arr[0];  // Access the first element (1)
```

- Modify an element by assigning a new value to a specific index.

```
arr[0] = 10;  // Update the first element to 10
```

3. Looping through Arrays

Loops can be used to iterate over arrays to perform operations on each element.

- **Regular for-loop**:

```cpp
for (int i = 0; i < 5; i++)
{
    cout << arr[i] << " ";
}
```

- **Range-based for-loop** (C++11 and later):

```cpp
for (int value : arr)
{
    cout << value << " ";
}
```

4. Arrays as Function Parameters

Arrays can be passed to functions as parameters by specifying the array type and size (if known). Arrays are **passed by reference** only (note that the symbol & is not used as a regular parameter), meaning changes within the function **affect** the original array.
For example:

- Print Array Function:

```cpp
void printArray(int arr[], int size)
{
    for (int i = 0; i < size; i++)
        cout << arr[i] << " ";

    cout << endl;
}
```

- Input Array Function:

```cpp
void inputArray(int arr[], int &size)
{
    cin >> size;

    for (int i = 0; i < size; i++)
        cin >> arr[i];
}
```

5. Multidimensional Arrays

C/C++ allows Multidimensional Arrays, commonly used for 2D Arrays (matrices) or 3D Arrays. Bellow are some examples for 2D Array:

- **Declaration and Initialization**:

```cpp
int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

  Here, `matrix` is a 2x3 array, with 2 rows and 3 columns.

- **Accessing Elements in a 2D Array**:

```cpp
int x = matrix[0][1];
// Access element in the first row, second column (value: 2)
```

- **Modifying Elements in a 2D Array**:

```cpp
matrix[1][2] = 10;
// Updates the element in the second row, third column to 10
```

- **Loop through a 2D Array**:

```cpp
int rows, cols;

cin >> rows >> cols;

for (int i = 0; i < rows; i++)
    for (int j = 0; j < rows; j++)
        cin >> matrix[i][j];
```

- **Passing Multidimensional Array**: When passing a multidimensional array to a function, the sizes of all dimensions must be specified (maybe except the first dimension).

```cpp
void printMatrix(int matrix[][3], int rows)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}
```

Exercises

Exercise 1. Find the smallest even and largest odd number

Write a program to find the smallest even and largest odd number in an array.

Input:

- The number of elements in array - `n`.
- The array of integers.

Output:

- The smallest even and the largest odd number.
- If there is no even number, then print `No even number`.
- If there is no odd number, then print `No odd number`.

Example:

Input	**Output**

`5`
`1 2 3 4 5`
	`2`
	`5`

Exercise 2. Find the k^{th} largest element

Write a program to find the k^{th} largest element in an array.

Input:

- The number of elements in array - `n`; and the integer `k`.
- The array of integers.

Output:

- The k^{th} largest element (print `Not found` if there is no valid number).

Example:

Input	**Output**

`5 2`
`1 5 3 4 2`
	`4`

Exercise 3. Sort an array in ascending order

Write a program to sort an array in ascending order.

Input:

- The number of elements in array - `n`.
- The array of integers.

Output:

- The sorted array.

Example:

Input	**Output**

`5`
`3 1 2 5 4`
	`1 2 3 4 5`

Exercise 4. Delete number

Write a program to delete a number from an array.

Input:

- The number of elements in array - `n`; and the number `k` that need to be deleted.
- The array of integers.

Output:

- The array without the value `k`.
- Notes: If the array has no elements left after deletion, output `Empty`.

Example:

Input	**Output**

`11 4`
`4 4 1 4 2 4 3 4 5 4 4`
	`1 2 3 5`

Exercise 5. Reverse an array

Write a program to reverse an array.

Input:

- The number of elements in array - `n`.
- The array of integers.

Output:

- The reversed array.

Example:

Input	**Output**

`5`
`1 2 3 4 5`
	`5 4 3 2 1`

Exercise 6. Check array

Write a program to check if an array is not increasing, or not decreasing, or just a regular array.

Input:

- The number of elements in array - `n`.
- The array of integers.

Output:

- `Not increasing` if the array is not an increasing array.
- `Not decreasing` if the array is not a decreasing array.
- Else print `Regular`.

Example:

Input	**Output**

`4`
`1 2 3 4`
	`Not decreasing`

Exercise 7. Find the longest non-decreasing sub-array

Write a program to find the longest non-decreasing sub-array in an array.

Input:

- The number of elements in array - `n`.
- The array of integers.

Output:

- The longest non-decreasing sub-array.
- If there are many results with the same length, print any.

Example:

Input	Output

`6`
`5 3 4 8 6 7`
	`3 4 8`

Exercise 8. Find the maximum sum sub-array

Write a program to find the sub-array with the largest sum.

Input:

- The number of elements in array - `n`.
- The array of integers.

Output:

- The maximum sum sub-array.
- If there are many results, print any.

Example:

Input	Output

`9`
`-2 1 -3 4 -1 2 1 -5 4`
	`4 -1 2 1`

Exercise 9. Histogram of an array

Write a program to generate a histogram representation of an array.

Input:

- The number of elements in array - `n`.
- The array of integers.

Output (Sorted in ascending order):

- The number: its number of occurrences.

Example:

Input	Output

`11`
`3 2 1 1 1 2 1 1 3 2 2`
	`1:  5`
	`2:  4`
	`3:  2`

Exercise 10. Merge two arrays

Write a program to merge two arrays that already sorted in ascending order to an ascending array.

Input:

- The number of elements in the first array - `n`, and the second array - `m`.
- The first array of integers.
- The second array of integers.

Output:

- The merged ascending array.

Example:

Input	Output

`2 2`
`1 3`
`2 4`
	`1 2 3 4`

Exercise 11. Find smallest and largest prime in a matrix

Write a program to find the smallest and largest prime numbers in a matrix.

Input:

- The number of rows `n` and the number of columns `m`.
- The matrix where each row is on a line and each element is separated by a space.

Output:

- The smallest and largest prime in the matrix.

Example:

Input	**Output**

`2 2`
`2 4`
`5 6`
	`2 5`

Exercise 12. Sort a matrix in ascending order

Write a program to sort all elements of a matrix in ascending order.

Input:

- The number of rows `n` and the number of columns `m`.
- The matrix where each row is on a line and each element is separated by a space.

Output:

- The sorted matrix.

Example:

Input	**Output**

`2 2`
`3 1`
`4 2`
	`1 2`
	`3 4`

Exercise 13. Delete a row from a matrix

Write a program to delete a specific row from a matrix.

Input:

- The number of rows `n`, the number of columns `m` and the number `k`.
- The matrix where each row is on a line and each element is separated by a space.

Output:

- The matrix without the row `k`.
- Notes: If the matrix has no elements left after deletion, output `Empty`.

Example:

Input	Output

`2 2 2`
`1 2`
`3 4`
	`1 2`

Exercise 14. Delete a column from a matrix

Write a program to delete a specific column from a matrix.

Input:

- The number of rows `n`, the number of columns `m` and the number `k`.
- The matrix where each row is on a line and each element is separated by a space.

Output:

- The matrix without the column `k`.
- Notes: If the matrix has no elements left after deletion, output `Empty`.

Example:

Input	Output

`2 2 2`
`1 2`
`3 4`
	`1`
	`3`

Exercise 15. Rotate a matrix

Write a program to rotate a matrix 90 degrees clockwise.

Input:

- The number of rows `n`, the number of columns `m`.
- The matrix where each row is on a line and each element is separated by a space.

Output:

- The rotated matrix.

Example:

Input	**Output**

`2 2`
`1 2`
`3 4`
	`3 1`
	`4 2`

The end.