Data Structures Lab – Assignment 2

January 14, 2026

Topic: Multi-Dimensional Array Operations

Objective

To understand the concept of one-dimensional and multi-dimensional arrays. Develop the ability to create and manipulate arrays and matrices, perform conversions between one-dimensional (1-D) and two-dimensional (2-D) data structures, and apply algorithmic techniques such as in-place rearrangement and the two-pointer method.

Instructions

- Programs must use arrays only (no advanced data structures).
- Input size should be user-defined and not hard-coded.
- Display clear input and output as per the sample test case.
- Use proper indentation and comments.
- Language: Choose as per your interest.
- Ensure that the **execution time** of each code is saved for later use.

Q1. Creation of a 2-D Matrix with Random Integers

Problem Statement: Create a 2-D matrix of dimensions $N \times M$. Randomly assign integer values to each cell of the matrix such that each value lies between -10 and $+10$ (inclusive). Save this matrix as Mat.

Input:

- Two integers N and M representing the number of rows and columns.

Output:

- A 2-D matrix Mat of size $N \times M$ with integer elements in the range $[-10, 10]$.

Sample Input:

```
N = 3, M = 4
```

Sample Output:

```
 2  -5   7   0
-3   9  -1   4
 6  -8   1  -2
```

—

Q2. Conversion of a 1-D Array into a 2-D Matrix

Problem Statement: Given a one-dimensional array of size $1 \times NM$, convert it into a two-dimensional matrix of dimensions $N \times M$.

Input:

- Two integers N and M
- A 1-D array of size $1 \times NM$

Output:

- A 2-D matrix of dimensions $N \times M$

Sample Input:

```
N = 2, M = 3
Array = [1, 2, 3, 4, 5, 6]
```

Sample Output:

```
1  2  3
4  5  6
```

—

Q3. Conversion of a 2-D Matrix into a 1-D Array

Problem Statement: Given a two-dimensional matrix of size $N \times M$, convert it into a one-dimensional array by traversing the matrix row-wise.

Input:

- Two integers N and M
- A 2-D matrix of size $N \times M$

Output:

- A 1-D array containing all elements of the matrix

Sample Input:

```
N = 2, M = 3
Matrix =
1  2  3
4  5  6
```

Sample Output:

```
[1, 2, 3, 4, 5, 6]
```

—

Q4. Rearranging Array Elements Using an Auxiliary Array

Problem Statement: Given an integer array A, write a program to move all even numbers to the left side of the array and all odd numbers to the right side. **Note** that the order of elements of the resulting array should not be changed from the input array. **Hint:** Use additional buffer array(s) to store intermediate results.

Input:

- An integer N
- An integer array A of size N

Output:

- The rearranged array with even numbers on the left and odd numbers on the right

Test Cases:

1. **Sample Input:**

   ```
   N = 6
   A = [3, 8, 5, 12, 7, 6]
   ```

   Sample Output:

   ```
   [8, 12, 6, 3, 5, 7]
   ```

2. **Sample Input:**

   ```
   N = 6
   A = [8, 3, 5, 7, 11, 6]
   ```

   Sample Output:

   ```
   [8, 6, 3, 5, 7, 11]
   ```

—

Q5. Pattern Matching in a Two-Dimensional Array

You are given a two-dimensional grid of size $n \times n$ consisting of lower-case English letters. Your task is to determine how many times the word ‘‘saba’’ appears in the grid.

The word ‘‘saba’’ may appear in any of the following directions:

- Horizontally (left to right)
- Vertically (top to bottom)
- Diagonally (top-left to bottom-right)

Input Format

- The first line contains a single integer n ($1 \leq n \leq 100$), representing both the number of rows and columns of the grid.
- The next n lines each contain a string of length n consisting of lower-case English letters.

Output Format

- Print a single integer representing the total number of occurrences of the word ‘‘saba’’ in the grid.

Sample Input

```
5
sabab
aabsa
babsa
ababa
sabba
```

Sample Output

```
3
```

Explanation

The word ‘‘saba’’ appears:

- Once horizontally
- Once vertically
- Once diagonally

Hence, the total number of occurrences is 3.

—

*Q6. In-place Rearrangement of Even and Odd Elements

Problem Statement: Given an integer array A, rearrange the array such that all even numbers appear on the left and all odd numbers appear on the right. **Note** that the order of elements of the resulting array should not be changed from the input array. **Important Rule:** The rearrangement must be performed *in-place*, i.e., without using any extra array or additional memory (except for a few variables).

Input:

- An integer N
- An integer array A of size N

Output:

- The array after in-place rearrangement

Test Cases:

1. **Sample Input:**

   ```
   N = 6
   A = [3, 8, 5, 12, 7, 6]
   ```

   Sample Output:

   ```
   [8, 12, 6, 3, 5, 7]
   ```

2. **Sample Input:**

   ```
   N = 6
   A = [8, 3, 5, 7, 11, 6]
   ```

   Sample Output:

   ```
   [8, 6, 3, 5, 7, 11]
   ```

—

Hint: Solve this problem using 2-pointer technique

—