Data Structures and Algorithms

Lab 01 - Arrays

Exercises/Tasks:

1. Write a program to copy the elements of one array into another array **but in reverse order**. For example, if the first array contains the values 1, 2, 3, 4, 5 then the second array should contain the values (after copying from the first array) 5, 4, 3, 2, 1.

   Note: Take the size of the first array as input from the user, then get those much values from the user, store them in the first array, and print the first array's values. Then copy the values from the first array, store them in reverse order in the second array, and print the second array's values.

2. Create a program that involves copying elements from two different arrays into a third array. Initially, the program should prompt the users to input the sizes of both arrays. Subsequently, the program should request the user to input the corresponding number of values for each array. The program should then store these values in the respective arrays. Following this, the program should display the values of both arrays separately.

   Following the display of the arrays, the program should proceed to copy the elements from the first and second arrays into a third array. It is important to note that the sizes of the first and second arrays can vary, implying that they may not necessarily be of the same size. During the copying process into the third array, you should first copy the values from the first array into the lower indices of the third array, and subsequently store the values of the second array in the remaining (upper) indices of the third array. Finally, the program should display the elements of the third array to demonstrate that the values have been copied accurately.

   Sample Output:

   ```
   Enter the size of the first array: 3
   Enter value 1: 5
   Enter values 2: 34
   Enter value 3: 12

   Enter the size of the second array: 4
   Enter value 1: 23
   ```

```
Enter values 2: 8
Enter value 3: 45
Enter value 4: 27

The combined elements from array 1 and array 2 stored in array 3 are:
5, 34, 12, 23, 8, 45, 27
```

3. Write a program to count (and print) the number of duplicate values in an array. You can store the fixed (hard-coded) values in the array. Then print those array values. Finally, count and print the number of duplicate values.

   Sample Output:

   ```
   Array values are:
   21, 9, 17, 43, 9, 15, 21, 6, 38, 6

   The count of duplicate values is: 3
   ```

4. Write a program that removes the duplicate values from an array. First print the original array (you can store the hard-coded values). Then, remove the duplicate values from that array. Finally, print the new array (obtained after removing the duplicate values), and print the sizes of both the original array and the new array.

   Sample Output:

   ```
   Original array is:
   21, 9, 17, 43, 9, 15, 21, 6, 38, 6

   Array after removing duplicates is:
   21, 9, 17, 43,15, 6, 38

   The size of the original arrays was: 10
   The size of the new arrays is: 7
   ```

5. Write a program to count (and print) the frequency of each element of the array (means, how many times each element occurs in the array). You can store the fixed (hard-coded) values in the array. Then print those array values. Finally, count and print the frequency of each element.

   Sample Output:

   ```
   Array values are:
   21, 9, 17, 9, 21, 9

   The frequency of all elements of the array are:
   21 occurs 2 times
   9 occurs 3 times
   17 occurs 1 times
   ```

6. Write a program that checks if a 2D array is **row-magic** (which means that every row has the same row sum). If the sum of each of the rows in the entire 2D array is the same, print “Row Magic” else print “Array is not Row-Magic”. First, print the original array elements and then check (and print) if the array is Row Magic or not

   Case 1 Sample Output:

   ```
   Array values are:
   4       7       2
   2       6       5
   9       3       1

   The array is Row-Magic
   ```

   Case 2 Sample Output:

   ```
   Array values are:
   3       7       2
   8       4       1
   5       9       6

   The array is not Row-Magic
   ```

7. Write a program that checks if a 2D array is **column-magic** (which means that every column has the same column sum). If the sum of each of the columns in the entire 2D array is the same, print “Column Magic” else print “Array is not Column-Magic”. First, print the original array elements and then check (and print) if the array is Column Magic or not.

   Case 1 Sample Output:

   ```
   Array values are:
   4       2       9
   7       6       3
   2       5       1

   The array is Column-Magic
   ```

   Case 2 Sample Output:

   ```
   Array values are:
   3       8       5
   7       4       9
   2       1       6

   The array is not Column-Magic
   ```

8. Write a program that flips a 2D array horizontally. First, print the original array and then print the flipped version of that array.

   Sample Output:

   ```
   Original array values are:
   3       7       2
   8       4       1
   5       9       6

   Horizontally flipped array is
   2       7       3
   1       4       8
   6       9       5
   ```

9. Write a program that flips a 2D array vertically. First, print the original array and then print the flipped version of that array.

Sample Output:

```
Original array values are:
3       7       2
8       4       1
5       9       6

Vertically flipped array is
5       9       6
8       4       1
3       7       2
```