Array-Based C Lab Questions

Instructions for Completing the Array-Based C Lab Questions

Objective: Implement the provided array-based C lab questions and save each program as a separate text document.

Deadline:

- **Date:** October 30, 2024
- **Time:** Before 5 PM

Naming Convention

For the "Shift Array Elements" question, save your C program using the following naming convention:

- For question 1: `shift_array_elements.c`

Ensure that your program compiles and produces the expected output before saving it.

Submission Instructions

- Submit all saved text documents by the deadline mentioned above.
- Follow your instructor's submission guidelines for how to submit the files.

Important: Make sure to test your program thoroughly to confirm that it works as intended. Late submissions may not be accepted, so manage your time effectively.

If you have any questions or need assistance, feel free to reach out.

Array-Based C Lab Questions

1. **Shift Array Elements:** Write a C program that takes an array of integers as input and shifts all elements to the right by a specified number of

positions. Fill the leftmost positions with zeros. For example, if the input array is {1, 2, 3, 4, 5} and the shift count is 2, the output should be {0, 0, 1, 2, 3}.

Expected Output:

```
Input: Array: {1, 2, 3, 4, 5}, Shift: 2
Output: {0, 0, 1, 2, 3}
```

2. **Identify the Largest Value:** Develop a function in C that scans through an array of integers to identify both the largest value and its corresponding index. The program should take an array like {10, 50, 30, 20} and output the largest value along with its index.

   Expected Output:

   ```
   Input: Array: {10, 50, 30, 20}
   Output: Largest Value: 50, Index: 1
   ```

3. **Combine Two Sorted Lists:** Create a C program that merges two sorted arrays into one sorted array without using any additional space. The program should take two sorted arrays as input and produce a combined sorted array.

   Expected Output:

   ```
   Input: Array1: {1, 3, 5}, Array2: {2, 4, 6}
   Output: Combined Array: {1, 2, 3, 4, 5, 6}
   ```

4. **Symmetry Check:** Write a C function to determine whether an array is symmetric, meaning it should read the same forwards and backwards. The function should handle cases like {5, 3, 5, 3, 5}.

   Expected Output:

   ```
   Input: Array: {5, 3, 5, 3, 5}
   Output: The array is symmetric.
   ```

5. **Element Occurrences:** Implement a C program that counts how many times each unique number appears in a given array. For example, for the input {1, 1, 2, 3, 2}, the program should count occurrences of each number.

   Expected Output:

   ```
   Input: Array: {1, 1, 2, 3, 2}
   Output: 1: 2, 2: 2, 3: 1
   ```

6. **Filter Unique Values:** Write a C program to filter out duplicate values from an array, returning a new array that contains only unique values and its new length. The input might be {1, 1, 2, 3, 3}, and the output should be a unique array.

   Expected Output:

   ```
   Input: Array: {1, 1, 2, 3, 3}
   Output: Unique Array: {1, 2, 3}, New Length: 3
   ```

7. **Maximum Sum of Non-Contiguous Elements:** Create a C function that calculates the maximum sum of non-contiguous elements in an array. For example, given the array {3, 2, 7, 10}, find the maximum sum that can be formed.

   Expected Output:
   ```
   Input: Array: {3, 2, 7, 10}
   Output: Maximum Sum: 13
   ```

8. **Common Elements Finder:** Write a C program that identifies and displays common elements between two distinct arrays. For input arrays like {1, 2, 3, 4} and {2, 3, 5, 6}, the program should output the common elements.

   Expected Output:
   ```
   Input: Array1: {1, 2, 3, 4}, Array2: {2, 3, 5, 6}
   Output: Common Elements: {2, 3}
   ```

9. **90-Degree Matrix Rotation:** Implement a C function that rotates a given 2D array (matrix) by 90 degrees to the right. The function should take a matrix as input and display the rotated version.

   Expected Output:
   ```
   Input:
   1 2 3
   4 5 6
   7 8 9
   Output:
   7 4 1
   8 5 2
   9 6 3
   ```

10. **Find the Missing Integer:** Create a C program to find the missing integer in a continuous range from ‘1‘ to ‘n‘, given an array that contains some of these integers. For example, if the input is {1, 2, 4, 5}, the output should indicate the missing integer.

    Expected Output:
    ```
    Input: Array: {1, 2, 4, 5}, n = 5
    Output: Missing Integer: 3
    ```

11. **Select Kth Smallest Item:** Write a C function that locates the Kth smallest number in an unordered array. The program should be able to take any array and return the Kth smallest element.

    Expected Output:
    ```
    Input: Array: {7, 10, 4, 3, 20, 15}, k = 4
    Output: 4th Smallest Element: 10
    ```

12. **Contiguous Subarray Finder:** Design a C program that searches for a contiguous subarray that sums up to a specified target value. The input array and the target should be provided to find the required subarray.

Expected Output:

```
Input: Array: {2, 4, 1, 5, 6}, Target Sum: 11
Output: Subarray found from index 1 to 4
```

13. **Spiral Matrix Printing:** Create a C function to display the elements of a matrix in spiral order. The function should handle square matrices and print the elements as they appear in a spiral pattern.

    Expected Output:

    ```
    Input:
    1 2 3
    4 5 6
    7 8 9
    Output: 1 2 3 6 9 8 7 4 5
    ```

14. **Alternating Sequence:** Write a C program that rearranges an array so that it alternates between positive and negative values. The program should take an array as input and output the rearranged version.

    Expected Output:

    ```
    Input: Array: {-1, 3, -2, 4, -5}
    Output: Rearranged Array: {3, -1, 4, -2, -5} or similar
    ```

15. **Count Array Inversions:** Implement a C function that counts the number of inversions in an array. An inversion is defined as a pair of indices such that the element at the first index is greater than the element at the second index.

    Expected Output:

    ```
    Input: Array: {3, 1, 2}
    Output: Number of Inversions: 2
    ```

16. **Consecutive Sequence Length:** Write a C program that finds the length of the longest sequence of consecutive integers in a given array. The input might be {100, 4, 200, 1, 3, 2}.

    Expected Output:

    ```
    Input: Array: {100, 4, 200, 1, 3, 2}
    Output: Length of Longest Consecutive Sequence: 4
    ```

17. **Element-Wise Addition:** Create a C program that performs element-wise addition of two arrays of the same length. The program should take two arrays and return their sum.

    Expected Output:

    ```
    Input: Array1: {1, 2, 3}, Array2: {4, 5, 6}
    Output: Resulting Array: {5, 7, 9}
    ```

18. **Transpose a Given Matrix:** Write a C function to transpose a square matrix and display the result. The input matrix should be processed to produce its transpose.

Expected Output:

```
Input:
1 2 3
4 5 6
7 8 9
Output:
1 4 7
2 5 8
3 6 9
```

19. **Longest Increasing Series:** Write a C program that determines the length of the longest increasing series in a given array. The program should be able to handle various input arrays and return the length.

    Expected Output:

    ```
    Input: Array: {10, 22, 9, 33, 21, 50, 41, 60, 80}
    Output: Length of Longest Increasing Subsequence: 6
    ```

20. **Dynamic Array Operations:** Write a C program to dynamically handle an array with operations such as adding and removing elements. The program should maintain the array's integrity during these operations.

    Expected Output:

    ```
    Input: Initial Array: {1, 2, 3}, Add: 4, Remove: 2
    Output: Final Array: {1, 3, 4}
    ```

21. **Maximize Non-Adjacent Sum:** Implement a function to calculate the maximum possible sum of non-adjacent elements from an array. The program should efficiently compute the sum without including adjacent elements.

    Expected Output:

    ```
    Input: Array: {3, 2, 5, 10, 7}
    Output: Maximum Non-Adjacent Sum: 15
    ```

22. **Longest Shared Prefix:** Write a C program to find the longest common prefix among an array of strings. The input should consist of multiple strings, and the output should be the longest shared prefix.

    Expected Output:

    ```
    Input: Array: {"flower", "flow", "flight"}
    Output: Longest Common Prefix: "fl"
    ```

23. **Generate a Triangle:** Create a C function to generate the first 'n' rows of a triangular number pattern. The output should display the triangular formation clearly.

    Expected Output:

    ```
    Input: n = 5
    Output:
    1
    ```

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

24. **Kth Largest Element with Heap:** Implement a C program to find the Kth largest element in an array using a heap data structure. The program should efficiently locate the specified element.

    Expected Output:

    ```
    Input: Array: {3, 2, 1, 5, 6, 4}, k = 2
    Output: 2nd Largest Element: 5
    ```

25. **Separate Even and Odd Elements:** Write a C program to segregate even and odd numbers from a given array into two separate arrays. The program should produce distinct arrays for evens and odds.

    Expected Output:

    ```
    Input: Array: {1, 2, 3, 4, 5, 6}
    Output: Even Array: {2, 4, 6}, Odd Array: {1, 3, 5}
    ```