DAY-1 TASK(TEAM-D)

1.Find the Number of Elements in an Array:

main.c | Share | Run | Output

```c
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50}; // Example array
    int n = sizeof(arr) / sizeof(arr[0]); // Calculate the number of
        elements

    printf("Number of elements in the array: %d\n", n);
    return 0;
}
```

```
Number of elements in the array: 5


=== Code Execution Successful ===
```

2. Delete an Element from an Array:

main.c | Share | Run | Output

```c
int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int size = 5;  // Current size of the array

    int indexToDelete = 2;  // Index of element to delete (index 2 =
        30)

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

    // Delete the element at index 2
    deleteElement(arr, &size, indexToDelete);

    printf("Array after deletion: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
```

```
Original array: 10 20 30 40 50
Array after deletion: 10 20 40 50


=== Code Execution Successful ===
```

3.Find Sum of Array Elements using Pointer:

main.c | Share | Run | Output

```c
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};  // Example array
    int sum = 0;  // Variable to store the sum
    int *ptr = arr;  // Pointer pointing to the first element of the array
    int size = sizeof(arr) / sizeof(arr[0]);  // Calculate the size of the array

    // Traverse the array using the pointer
    for (int i = 0; i < size; i++) {
        sum += *(ptr + i);  // Access the element using pointer arithmetic
    }

    // Print the result
    printf("Sum of array elements: %d\n", sum);

    return 0;
}
```

```
Sum of array elements: 150


=== Code Execution Successful ===
```

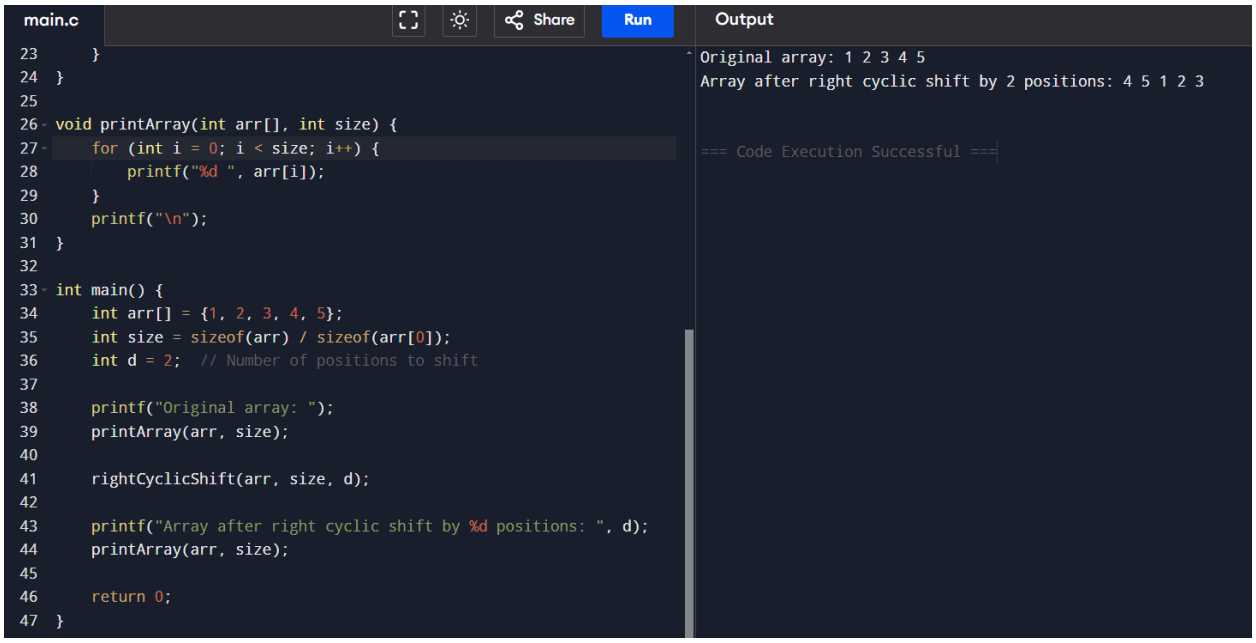
4.Print all Non Repeated Elements in an Array:

main.c | Share | Run | Output

```c
#include <stdio.h>

void printNonRepeatedElements(int arr[], int size) {
    int isRepeated;

    // Traverse each element of the array
    for (int i = 0; i < size; i++) {
        isRepeated = 0;

        // Check if the current element is repeated in the array
        for (int j = 0; j < size; j++) {
            // Skip comparing the element with itself
            if (i != j && arr[i] == arr[j]) {
                isRepeated = 1;
                break;  // No need to check further if a duplicate is found
            }
        }

        // If not repeated, print the element
        if (!isRepeated) {
            printf("%d ", arr[i]);
        }
    }
}
```

```
Non-repeated elements: 20 40 50

=== Code Execution Successful ===
```

5.Cyclically Permute the Elements of an Array:



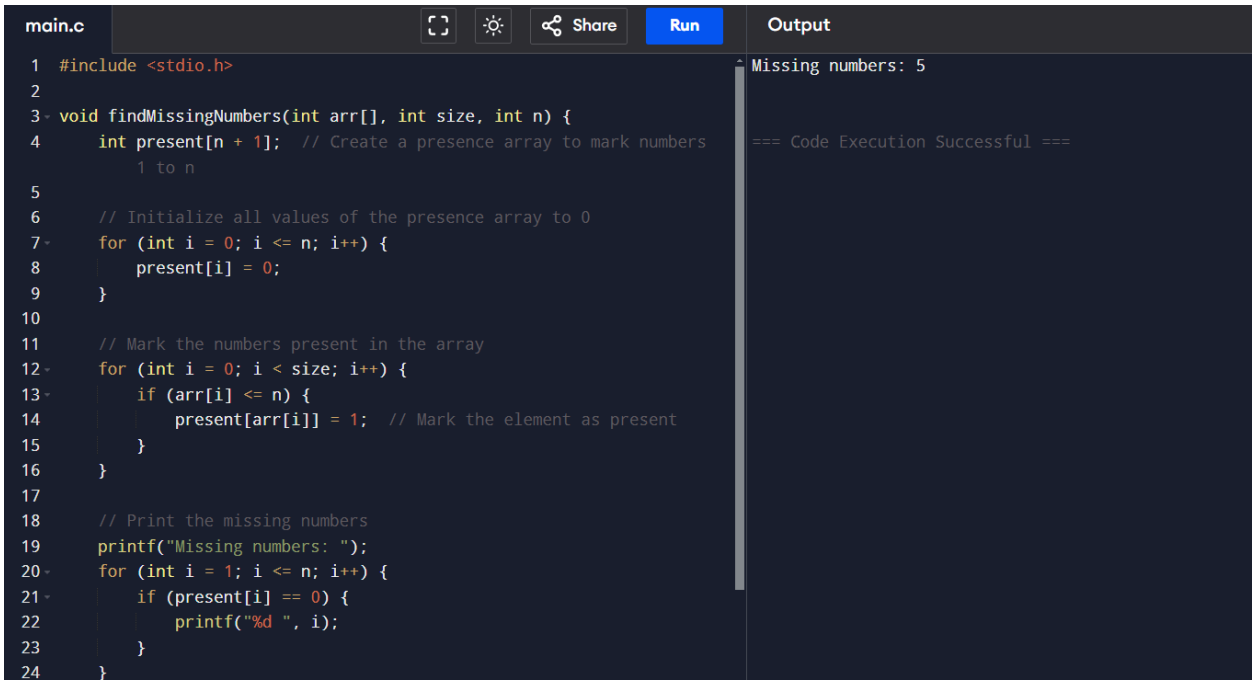
main.c

Share Run Output

```
    }
}

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

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    int d = 2;  // Number of positions to shift

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

    rightCyclicShift(arr, size, d);

    printf("Array after right cyclic shift by %d positions: ", d);
    printArray(arr, size);

    return 0;
}
```

```
Original array: 1 2 3 4 5
Array after right cyclic shift by 2 positions: 4 5 1 2 3


=== Code Execution Successful ===
```

6.Find Missing Numbers in Array:



main.c

Share Run Output

```
#include <stdio.h>

void findMissingNumbers(int arr[], int size, int n) {
    int present[n + 1];  // Create a presence array to mark numbers
        1 to n

    // Initialize all values of the presence array to 0
    for (int i = 0; i <= n; i++) {
        present[i] = 0;
    }

    // Mark the numbers present in the array
    for (int i = 0; i < size; i++) {
        if (arr[i] <= n) {
            present[arr[i]] = 1;  // Mark the element as present
        }
    }

    // Print the missing numbers
    printf("Missing numbers: ");
    for (int i = 1; i <= n; i++) {
        if (present[i] == 0) {
            printf("%d ", i);
        }
    }
```

```
Missing numbers: 5


=== Code Execution Successful ===
```

7. Find Union and Intersection of Two Arrays:

main.c | Share | Run | Output

```
                    further for arr2[j]
            }
        }
    }

    // Print the intersection array
    printf("Intersection: ");
    for (int i = 0; i < k; i++) {
        printf("%d ", intersectionArr[i]);
    }
    printf("\n");
}

int main() {
    int arr1[] = {1, 2, 3, 4, 5};  // Example array 1
    int arr2[] = {4, 5, 6, 7};     // Example array 2

    int size1 = sizeof(arr1) / sizeof(arr1[0]);
    int size2 = sizeof(arr2) / sizeof(arr2[0]);

    // Find and print union and intersection
    findUnion(arr1, size1, arr2, size2);
    findIntersection(arr1, size1, arr2, size2);

    return 0;
}
```

```
Union: 1 2 3 4 5 6 7
Intersection: 4 5


=== Code Execution Successful ===
```

8.Split the Array and Add First Part to the End:

main.c | Share | Run | Output

```
    }
}

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

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    int k = 2;  // Split position (index at which the array is split
        )

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

    splitAndMoveToEnd(arr, size, k);

    printf("Array after split and move to end: ");
    printArray(arr, size);

    return 0;
```

```
Original array: 1 2 3 4 5
Array after split and move to end: 3 4 5 1 2


=== Code Execution Successful ===
```

9.Matrix Multiplication:

main.c

Share Run

```c
#include <stdio.h>

// Function to multiply two matrices
void multiplyMatrices(int firstMatrix[][10], int secondMatrix[][10],
    int resultMatrix[][10], int rowFirst, int colFirst, int
    rowSecond, int colSecond) {
    // Initialize elements of matrix mult to 0
    for (int i = 0; i < rowFirst; i++) {
        for (int j = 0; j < colSecond; j++) {
            resultMatrix[i][j] = 0;
        }
    }

    // Multiply the first matrix with the second matrix
    for (int i = 0; i < rowFirst; i++) {
        for (int j = 0; j < colSecond; j++) {
            for (int k = 0; k < colFirst; k++) {
                resultMatrix[i][j] += firstMatrix[i][k] *
                    secondMatrix[k][j];
            }
        }
    }
}

```

Output

```
Enter rows and columns for the first matrix: 2 3
Enter elements of the first matrix:
1 2 3
4 5 6
Enter rows and columns for the second matrix: 3 2
Enter elements of the second matrix:
7 8
9 10
11 12
Resultant Matrix:
58 64
139 154


=== Code Execution Successful ===
```