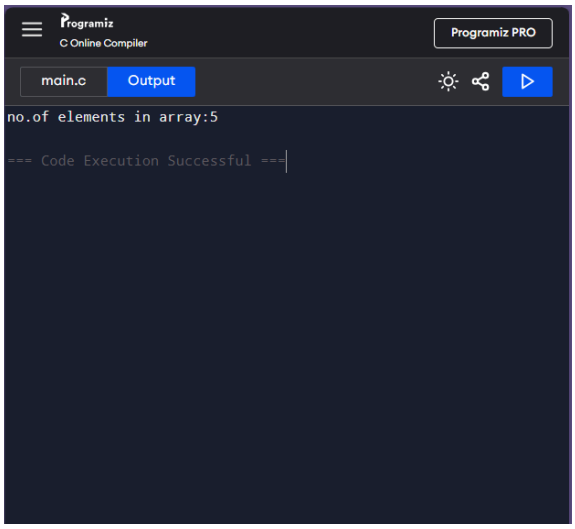
1. Find the Number of Elements in an Array

Code

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Number of elements in array: %d\n", n);
    return 0;
}
```

Output



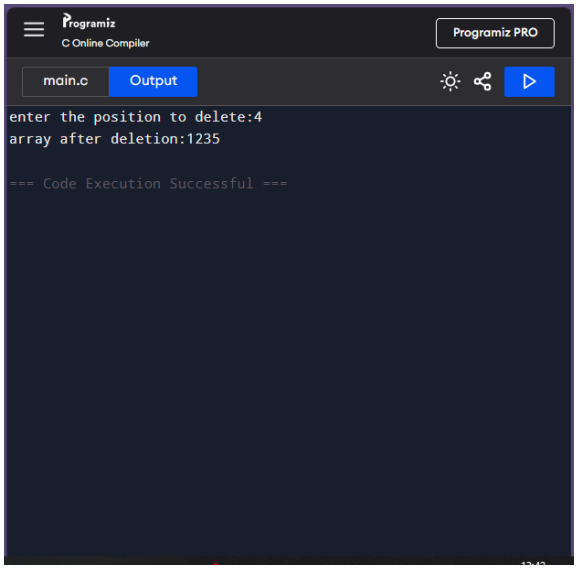
2. Delete an Element from an Array

Code

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    int pos;
    printf("Enter position to delete: ");
    scanf("%d", &pos);
    for(int i=pos-1; i<n-1; i++) {
        arr[i] = arr[i+1];
    }
    n--;
    printf("Array after deletion: ");
    for(int i=0; i<n; i++) {
```

```
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Output



3. Find Sum of Array Elements using Pointer

Code

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    int sum = 0;
    int *ptr = arr;
    for(int i=0; i<n; i++) {
        sum += *ptr;
        ptr++;
    }
    printf("Sum of array elements: %d\n", sum);
    return 0;
}
```

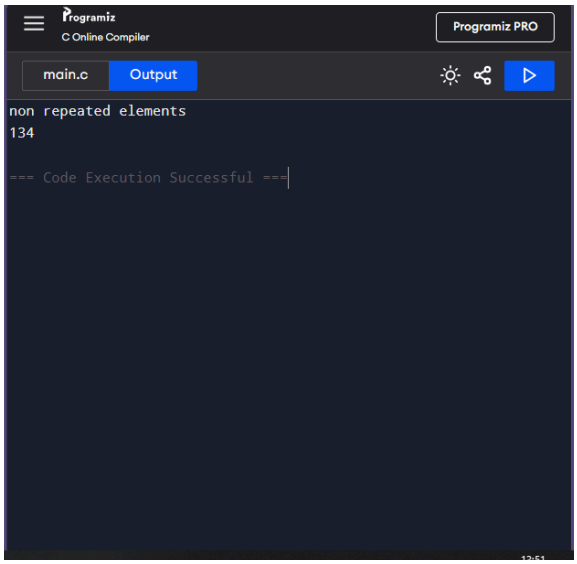
Output

4. Print all Non Repeated Elements in an Array

Code

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 2, 4, 5, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Non-repeated elements: ");
    for(int i=0; i<n; i++) {
        int count = 0;
        for(int j=0; j<n; j++) {
            if(arr[i] == arr[j]) count++;
        }
        if(count == 1) printf("%d ", arr[i]);
    }
    return 0;
}
```

Output

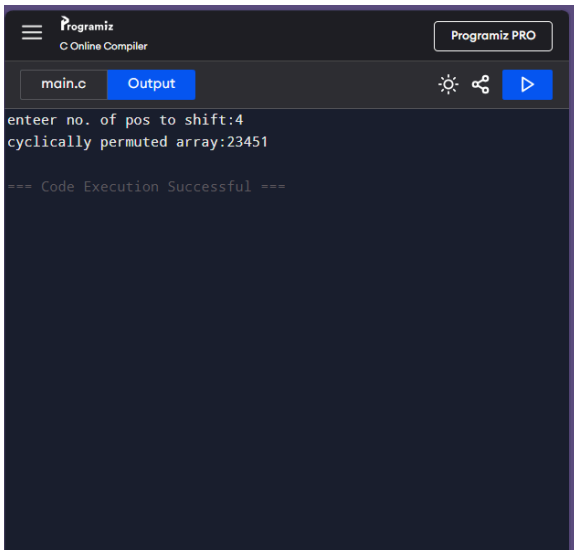


5. Cyclically Permute the Elements of an Array

Code

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    int k;
    printf("Enter number of positions to shift: ");
    scanf("%d", &k);
    printf("Cyclically permuted array: ");
    for(int i=n-k; i<n; i++) {
        printf("%d ", arr[i]);
    }
    for(int i=0; i<n-k; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Output

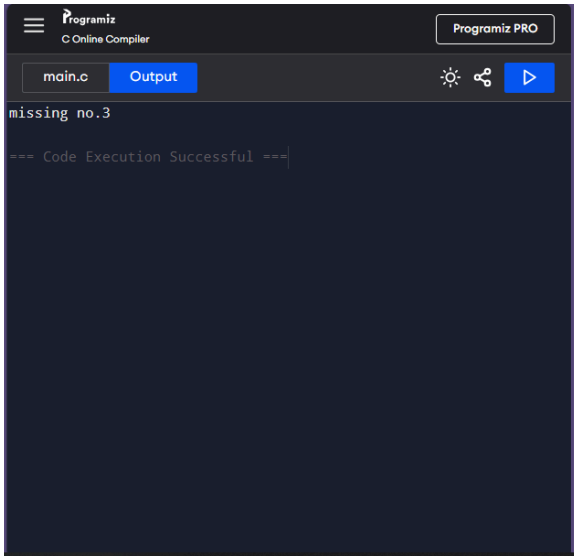


6. Find Missing Numbers in Array

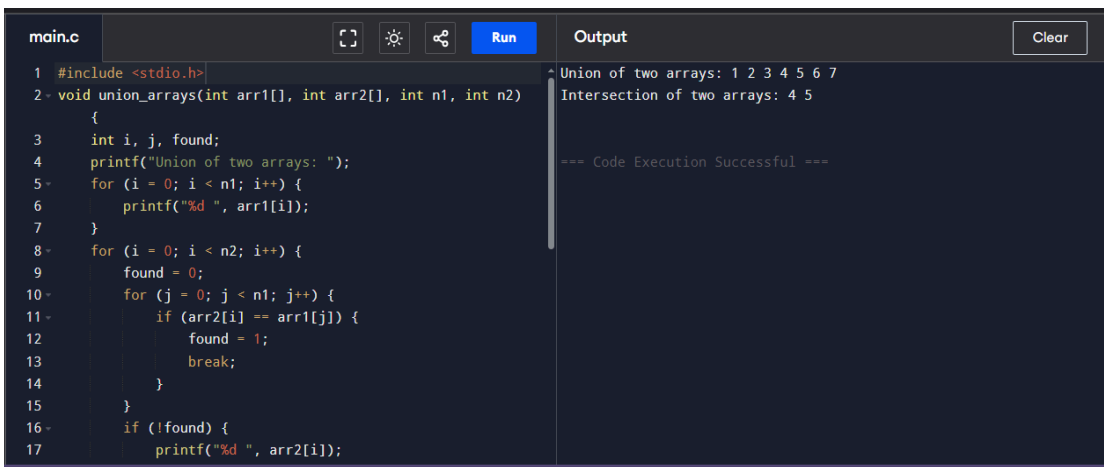
Code

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 4, 5, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Missing numbers: ");
    for(int i=1; i<=n+1; i++) {
        int found = 0;
        for(int j=0; j<n; j++) {
            if(arr[j] == i) {
                found = 1;
                break;
            }
        }
        if(!found) printf("%d ", i);
    }
    return 0;
}
```

Output



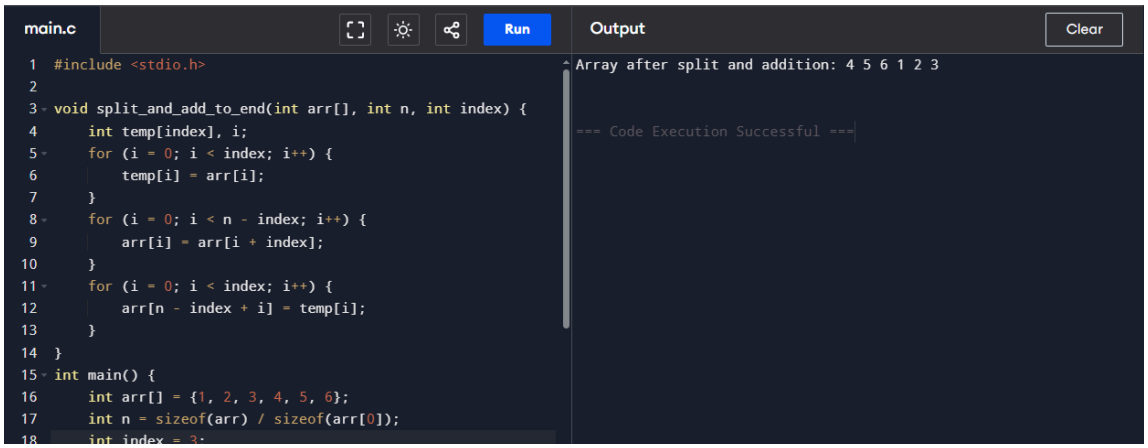
7. Find the Union and Intersection of Two Arrays




```c
#include <stdio.h>

void union_arrays(int arr1[], int arr2[], int n1, int n2) {
    int i, j, found;
    printf("Union of two arrays: ");
    for (i = 0; i < n1; i++) {
        printf("%d ", arr1[i]);
    }
    for (i = 0; i < n2; i++) {
        found = 0;
        for (j = 0; j < n1; j++) {
            if (arr2[i] == arr1[j]) {
```

```
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("%d ", arr2[i]);
    }
  }
  printf("\n");
}

void intersection_arrays(int arr1[], int arr2[], int n1, int n2) {
  int i, j, found;
  printf("Intersection of two arrays: ");
  for (i = 0; i < n1; i++) {
    for (j = 0; j < n2; j++) {
        if (arr1[i] == arr2[j]) {
            printf("%d ", arr1[i]);
            break;
        }
    }
  }
  printf("\n");
}

int main() {
  int arr1[] = {1, 2, 3, 4, 5};
  int arr2[] = {4, 5, 6, 7};
  int n1 = sizeof(arr1) / sizeof(arr1[0]);
  int n2 = sizeof(arr2) / sizeof(arr2[0]);

  union_arrays(arr1, arr2, n1, n2);
  intersection_arrays(arr1, arr2, n1, n2);

  return 0;
}
```

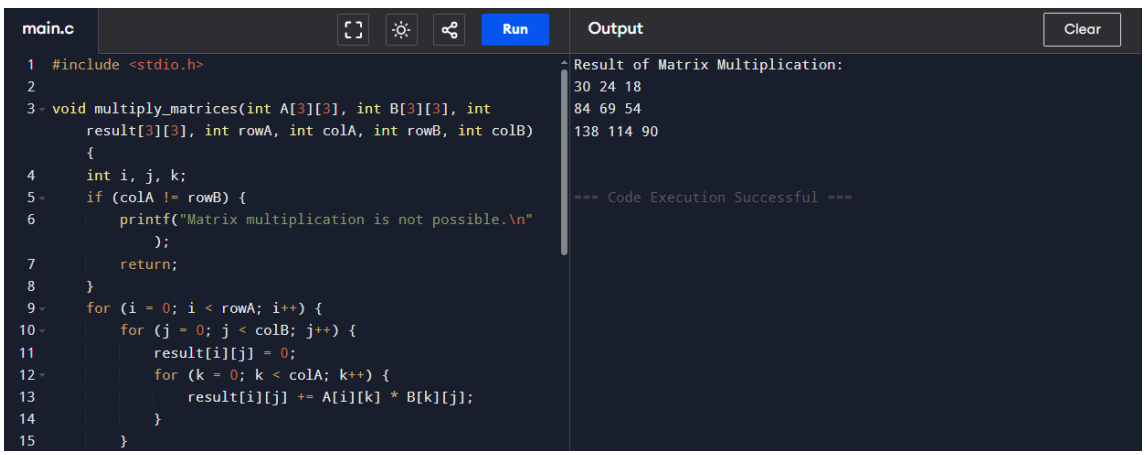
8. Split the Array and Add the First Part to the End




```
#include <stdio.h>

void split_and_add_to_end(int arr[], int n, int index) {
    int temp[index], i;

    // Copy the first part of the array to a temporary array
    for (i = 0; i < index; i++) {
        temp[i] = arr[i];
    }

    // Shift the elements in the original array
    for (i = 0; i < n - index; i++) {
        arr[i] = arr[i + index];
    }

    // Add the first part to the end of the original array
    for (i = 0; i < index; i++) {
        arr[n - index + i] = temp[i];
    }
}

int main() {
```

```
    int arr[] = {1, 2, 3, 4, 5, 6};
    int n = sizeof(arr) / sizeof(arr[0]);
    int index = 3; // Split at index 3

    split_and_add_to_end(arr, n, index);

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

    return 0;
}
```

9. Matrix Multiplication




```
#include <stdio.h>

void multiply_matrices(int A[3][3], int B[3][3], int result[3][3], int rowA, int colA, int rowB, int colB)
{
    int i, j, k;
    if (colA != rowB) {
        printf("Matrix multiplication is not possible.\n");
        return;
    }
    for (i = 0; i < rowA; i++) {
        for (j = 0; j < colB; j++) {
            result[i][j] = 0;
```

```
            for (k = 0; k < colA; k++) {
                result[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

int main() {
    int A[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int B[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
    int result[3][3];
    int rowA = 3, colA = 3, rowB = 3, colB = 3;

    multiply_matrices(A, B, result, rowA, colA, rowB, colB);

    printf("Result of Matrix Multiplication: \n");
    for (int i = 0; i < rowA; i++) {
        for (int j = 0; j < colB; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```