

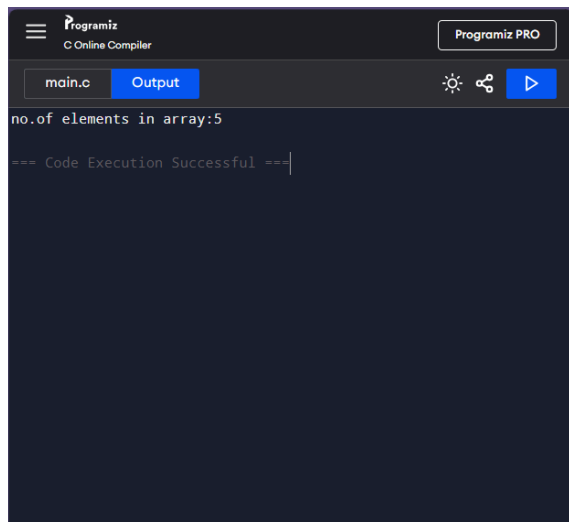
1. Find the Number of Elements in an Array

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

A screenshot of the Programiz Online Compiler interface. The top bar shows the Programiz logo and 'Programiz PRO' button. Below the bar, there's a tab labeled 'main.c' and a blue 'Output' button. The output area displays 'no.of elements in array:5' and a success message '== Code Execution Successful =='.

```
no.of elements in array:5  
  
== Code Execution Successful ==
```

2. Delete an Element from an Array

Code

```
#include <stdio.h>
```

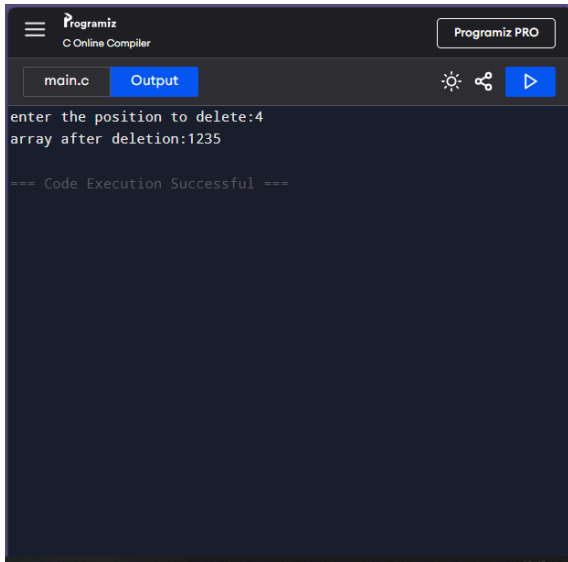
```
int main() {  
    int arr[] = {1, 2, 3, 4, 5};  
    int n = sizeof(arr)/sizeof(arr[0]);  
    int pos;  
    printf("Enter position to delete: ");  
    scanf("%d", &pos);  
    for(int i=pos-1; i<n-1; i++) {  
        arr[i] = arr[i+1];  
    }  
    n--;  
    printf("Array after deletion: ");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

```

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

```

Output



```

Programiz
C Online Compiler
Programiz PRO

main.c Output
enter the position to delete:4
array after deletion:1235

--- Code Execution Successful ---

```

3. Find Sum of Array Elements using Pointer

Code

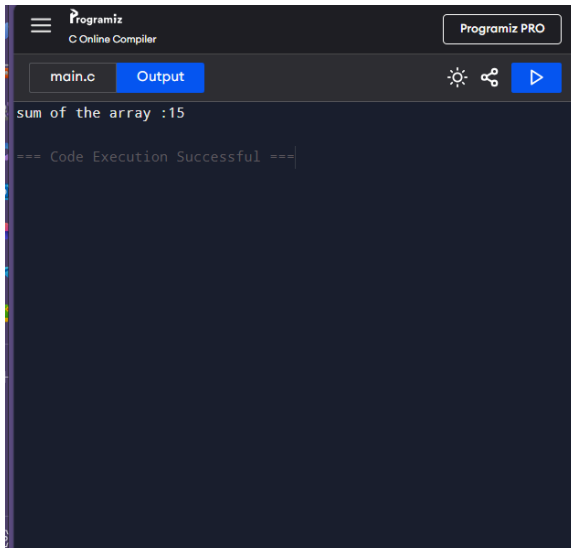
```
#include <stdio.h>
```

```

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

A screenshot of the Programiz C Online Compiler interface. The top bar shows the Programiz logo, 'C Online Compiler', and a 'Programiz PRO' badge. Below the bar, there are tabs for 'main.c' and 'Output'. The 'Output' tab is active, displaying the text 'sum of the array :15' and '=== Code Execution Successful ==='. The background is dark blue with a subtle grid pattern.

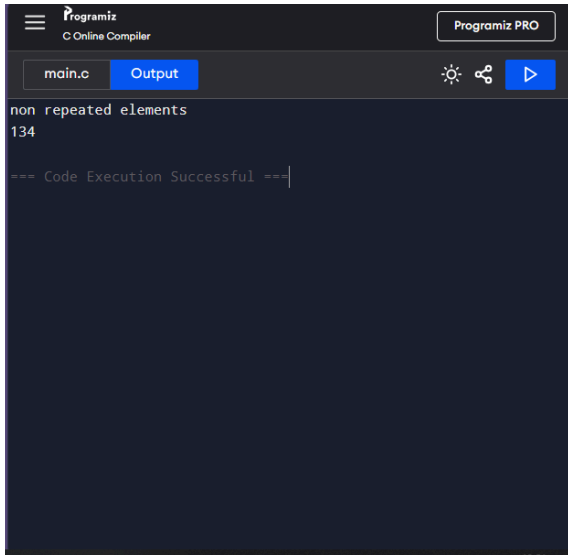
```
sum of the array :15
=== Code Execution Successful ===
```

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

The image shows a screenshot of the Programiz Online Compiler interface. At the top, there's a header with the Programiz logo, 'C Online Compiler', and a 'Programiz PRO' button. Below the header, there's a tab labeled 'main.c' and an 'Output' button. The output window displays the text 'non repeated elements' followed by the number '134'. At the bottom of the output window, it says '=== Code Execution Successful ==='.

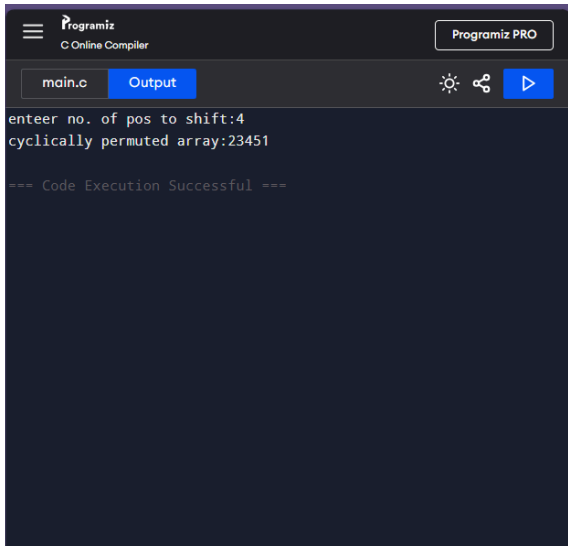
```
non repeated elements
134
=== Code Execution Successful ===
```

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

A screenshot of the Programiz C Online Compiler interface. The top bar shows the Programiz logo, 'C Online Compiler', and a 'Programiz PRO' badge. Below the bar, there are tabs for 'main.c' and 'Output'. The 'Output' tab is active, displaying the program's execution results. The output text reads: 'enter no. of pos to shift:4', 'cyclically permuted array:23451', and '=== Code Execution Successful ==='. The interface includes icons for settings, share, and run.

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

The screenshot shows the Programiz C Online Compiler interface. At the top, there's a header with the Programiz logo and 'C Online Compiler'. Below that, a tab labeled 'main.c' is active, and an 'Output' button is visible. The main editor area contains the text 'missing no.3' followed by '=== Code Execution Successful ==='. On the right side of the editor, there are icons for settings, a share icon, and a 'Run' button.

7. Find the Union and Intersection of Two Arrays

The screenshot shows the Programiz C Online Compiler interface with a C program. The code is as follows:

```
1 #include <stdio.h>
2 void union_arrays(int arr1[], int arr2[], int n1, int n2)
3 {
4     int i, j, found;
5     printf("Union of two arrays: ");
6     for (i = 0; i < n1; i++) {
7         printf("%d ", arr1[i]);
8     }
9     for (i = 0; i < n2; i++) {
10        found = 0;
11        for (j = 0; j < n1; j++) {
12            if (arr2[i] == arr1[j]) {
13                found = 1;
14                break;
15            }
16        }
17        if (!found) {
18            printf("%d ", arr2[i]);
19        }
20    }
21    printf("\n");
22 }
```

The output on the right shows: 'Union of two arrays: 1 2 3 4 5 6 7' and 'Intersection of two arrays: 4 5'. Below the output, it says '=== Code Execution Successful ==='. There is a 'Clear' button at the top right of the output area.

```
#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");
}

```

```

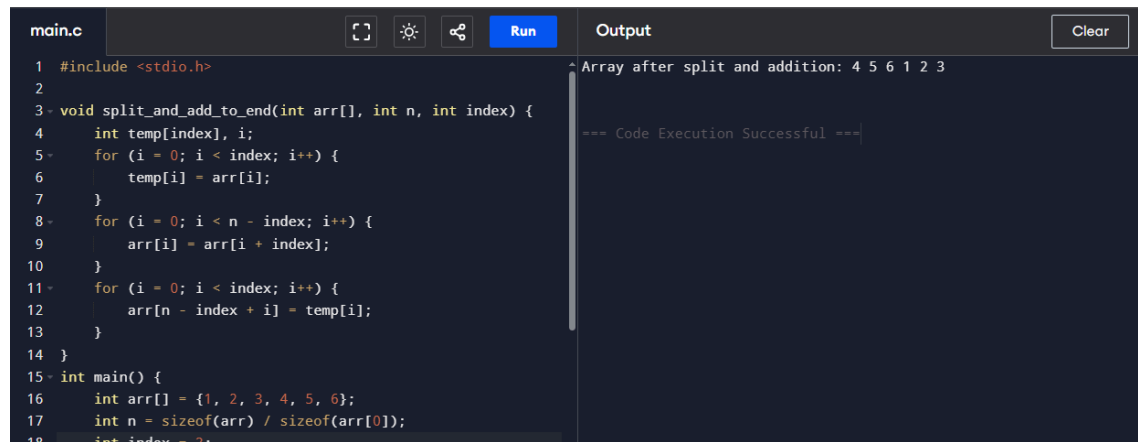
int main() {
    int arr1[] = {1, 2, 3, 4, 5};
    int arr2[] = {4, 5, 6, 7};
    int n1 = sizeof(arr1) / sizeof(arr1[0]);
    int n2 = sizeof(arr2) / sizeof(arr2[0]);

    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



```
main.c  [Icons]  Run  Output  Clear

1  #include <stdio.h>
2
3  void split_and_add_to_end(int arr[], int n, int index) {
4      int temp[index], i;
5      for (i = 0; i < index; i++) {
6          temp[i] = arr[i];
7      }
8      for (i = 0; i < n - index; i++) {
9          arr[i] = arr[i + index];
10     }
11     for (i = 0; i < index; i++) {
12         arr[n - index + i] = temp[i];
13     }
14 }
15 int main() {
16     int arr[] = {1, 2, 3, 4, 5, 6};
17     int n = sizeof(arr) / sizeof(arr[0]);
18     int index = 3;
```

Array after split and addition: 4 5 6 1 2 3

--- Code Execution Successful ---

```
#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

The screenshot shows a C code editor with a file named 'main.c'. The code implements a function 'multiply_matrices' that takes two 3x3 matrices A and B, and a 3x3 result matrix. It checks if the number of columns in A (colA) is equal to the number of rows in B (rowB). If not, it prints an error message and returns. Otherwise, it performs matrix multiplication using three nested loops: i for rows, j for columns, and k for the inner product. The output window shows the result of the multiplication: 30 24 18, 84 69 54, and 138 114 90. The code execution was successful.

```

main.c
1 #include <stdio.h>
2
3 void multiply_matrices(int A[3][3], int B[3][3], int
   result[3][3], int rowA, int colA, int rowB, int colB)
   {
4     int i, j, k;
5     if (colA != rowB) {
6         printf("Matrix multiplication is not possible.\n"
7             );
8         return;
9     }
10    for (i = 0; i < rowA; i++) {
11        for (j = 0; j < colB; j++) {
12            result[i][j] = 0;
13            for (k = 0; k < colA; k++) {
14                result[i][j] += A[i][k] * B[k][j];
15            }
16        }
17    }
18 }

```

Output

```

Result of Matrix Multiplication:
30 24 18
84 69 54
138 114 90

=== Code Execution Successful ===

```

```
#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;
}

```