

# DATA STRUCTURES

NAME : M. RUDRESH

REG NO : 192372339

COURSE CODE : CSA0312

TOPIC : ARRAYS

#program to merge two arrays

INPUT :-

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

```
void mergeArrays(int arr1[], int n1, int arr2[], int n2, int merged[]) {
```

```
    int i = 0, j = 0, k = 0;
```

```
    while (i < n1 && j < n2) {
```

```
        if (arr1[i] < arr2[j]) {
```

```
            merged[k++] = arr1[i++];
```

```
        } else {
```

```
            merged[k++] = arr2[j++];
```

```
        }
```

```
    }
```

```
    while (i < n1) {
```

```
        merged[k++] = arr1[i++];
```

```
    }  
    while (j < n2) {  
        merged[k++] = arr2[j++];  
    }  
}  
  
int main() {  
    int arr1[] = {1, 3, 5, 7};  
    int arr2[] = {2, 4, 6, 8};  
    int n1 = sizeof(arr1) / sizeof(arr1[0]);  
    int n2 = sizeof(arr2) / sizeof(arr2[0]);  
    int merged[n1 + n2];  
    mergeArrays(arr1, n1, arr2, n2, merged);  
    printf("Merged Array: ");  
    for (int i = 0; i < n1 + n2; i++) {  
        printf("%d ", merged[i]);  
    }  
  
    return 0;  
}
```

OUTPUT :-

## Output

/tmp/Rwj2bive3d.o

Merged Array: 1 2 3 4 5 6 7 8

=== Code Execution Successful ===

# program to find mini and maxi element in an array

INPUT :-

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

```
void findMinMax(int arr[], int n, int *min, int *max) {
```

```
    *min = arr[0];
```

```
    *max = arr[0];
```

```
    for (int i = 1; i < n; i++) {
```

```
        if (arr[i] < *min) {
```

```
            *min = arr[i];
```

```
        }
```

```
        if (arr[i] > *max) {
```

```
            *max = arr[i];
```

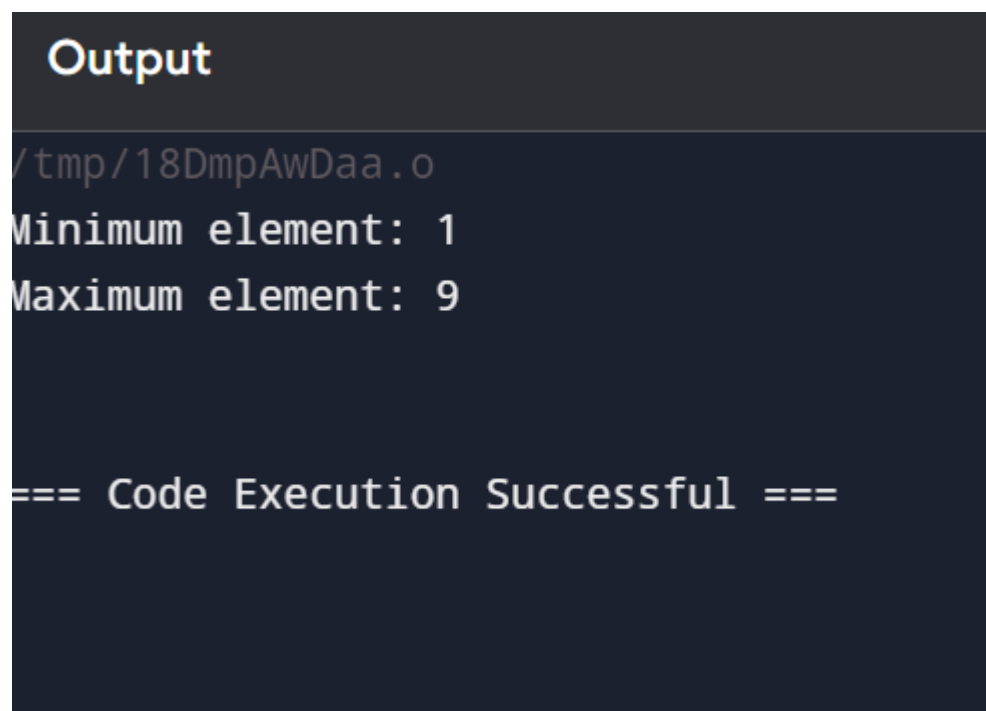
```
        }
```

```
    }
```

```
}
```

```
int main() {  
    int arr[] = {5, 3, 9, 1, 6, 7, 2};  
    int n = sizeof(arr) / sizeof(arr[0]);  
    int min, max;  
  
    findMinMax(arr, n, &min, &max);  
  
    printf("Minimum element: %d\n", min);  
    printf("Maximum element: %d\n", max);  
  
    return 0;  
}
```

## OUTPUT :-



The screenshot shows a dark-themed output window. At the top, the word "Output" is written in a light blue font. Below it, the file path "/tmp/18DmpAwDaa.o" is visible. The program's output is displayed in a light blue monospaced font: "Minimum element: 1" followed by "Maximum element: 9" on the next line. At the bottom of the window, the text "=== Code Execution Successful ===" is shown in the same light blue font.

```
Output  
/tmp/18DmpAwDaa.o  
Minimum element: 1  
Maximum element: 9  
  
=== Code Execution Successful ===
```

## #program to find a duplicate element in an array

### INPUT :-

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

```
void findDuplicates(int arr[], int n) {
```

```
    int foundDuplicate = 0;
```

```
    printf("Duplicate elements in the array: ");
```

```
    for (int i = 0; i < n - 1; i++) {
```

```
        for (int j = i + 1; j < n; j++) {
```

```
            if (arr[i] == arr[j]) {
```

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

```
                foundDuplicate = 1;
```

```
                break;
```

```
            }
```

```
        }
```

```
    }
```

```
    if (!foundDuplicate) {
```

```
        printf("No duplicates found.");
```

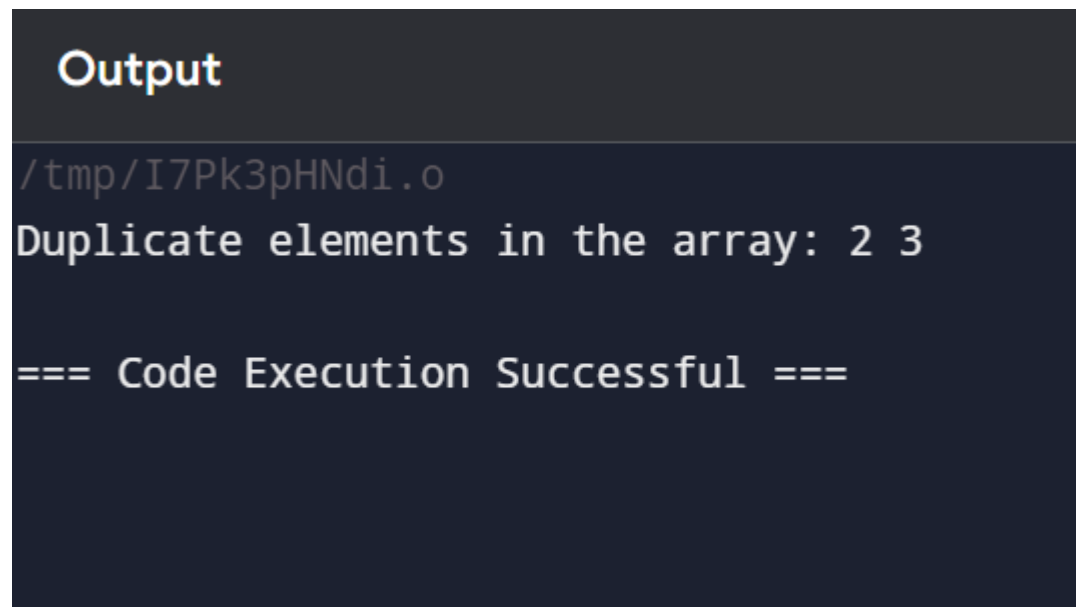
```
    }
```

```
}
```

```
int main() {
```

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

### OUTPUT :-

A screenshot of a code execution environment. At the top, the word "Output" is displayed in a light blue font. Below it, the file path "/tmp/I7Pk3pHNdi.o" is shown in a light gray font. The main output text, "Duplicate elements in the array: 2 3", is displayed in a light blue font. At the bottom, the status "=== Code Execution Successful ===" is shown in a light blue font.

```
Output  
/tmp/I7Pk3pHNdi.o  
Duplicate elements in the array: 2 3  
=== Code Execution Successful ===
```

### #program to sum of two arrays

#### INPUT :-

```
#include <stdio.h>  
  
void sumOfArrays(int arr1[], int arr2[], int n, int sum[]) {  
    for (int i = 0; i < n; i++) {  
        sum[i] = arr1[i] + arr2[i];  
    }  
}
```

```
    }  
}  
int main() {  
    int arr1[] = {1, 2, 3, 4};  
    int arr2[] = {5, 6, 7, 8};  
    int n = sizeof(arr1) / sizeof(arr1[0]);  
    if (sizeof(arr1) != sizeof(arr2)) {  
        printf("Error: Arrays must have the same size!\n");  
        return -1;  
    }  
    int sum[n];  
    sumOfArrays(arr1, arr2, n, sum);  
    printf("Sum of the two arrays: ");  
    for (int i = 0; i < n; i++) {  
        printf("%d ", sum[i]);  
    }  
  
    return 0;  
}
```

OUTPUT :-

## Output

```
/tmp/ymnLg3cgRb.o
```

```
Sum of the two arrays: 6 8 10 12
```

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