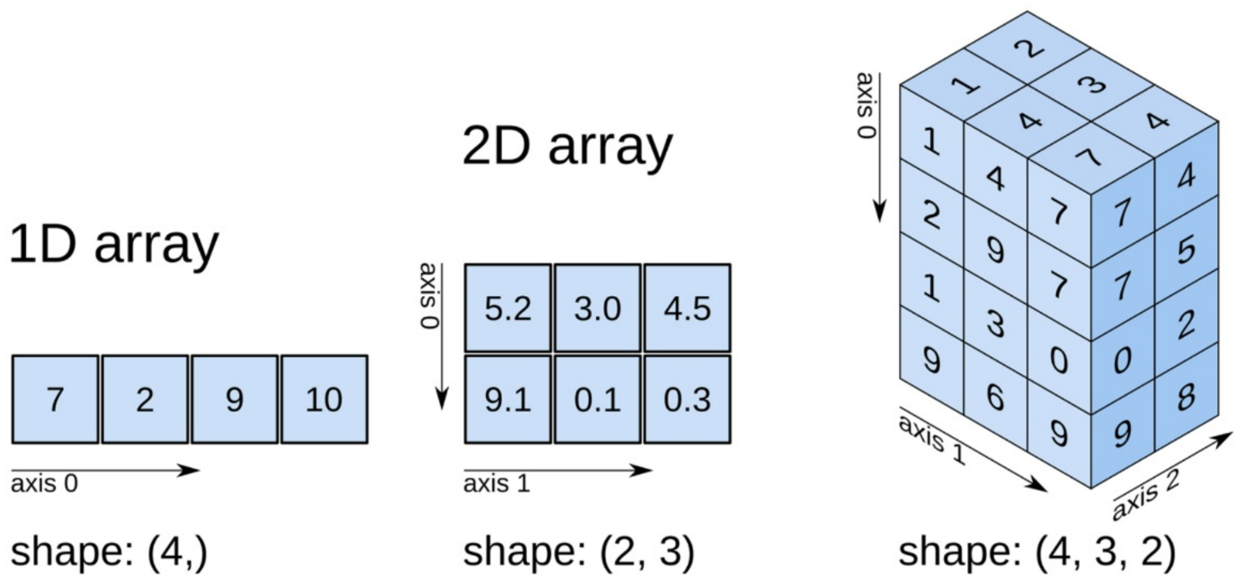
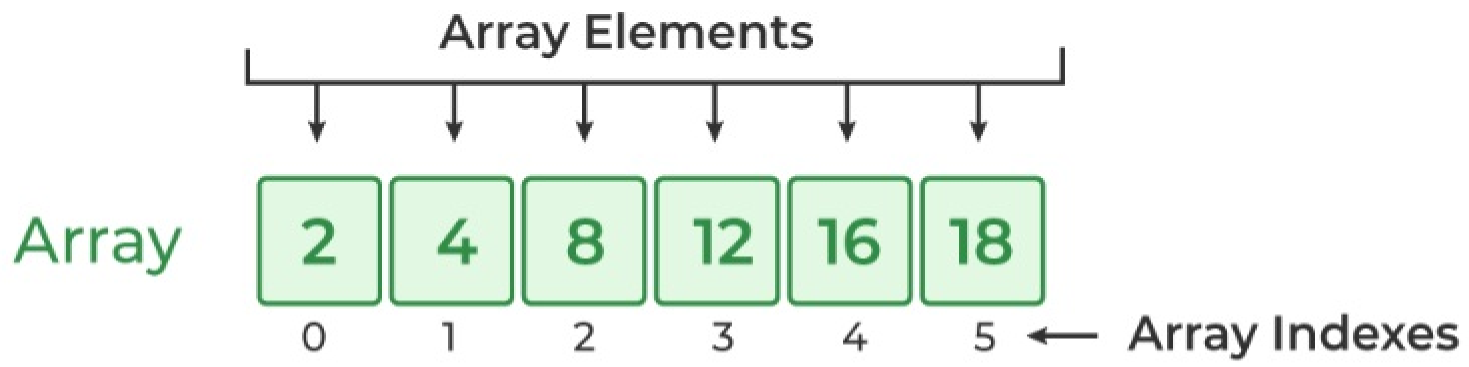


Implementation of 1D and Multi-Dimension Arrays


Array Elements
Array
2
4
8
12
16
18
0
1
2
3
4
5
Array Indexes

Problem 1: Insert Element at Index

Problem Statement:

Write a function to insert an element into a given array at a specified index.

```cpp
void insertAtIndex(int arr[], int &n, int value, int index);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Third line: Integer value to insert

Fourth line: Integer index where value is to be inserted

Output:

Print the array after insertion.

Sample Input:

```
5
1 2 3 4 5
10
2
```

Sample Output:

```
1 2 10 3 4 5
```

Problem 2: Delete Element at Index

Problem Statement:

Write a function to delete the element at a specified index from an array.

```cpp
void deleteAtIndex(int arr[], int &n, int index);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Third line: Integer index to delete

Output:

Print the array after deletion.

Sample Input:

```
6
10 20 30 40 50 60
3
```

Sample Output:

```
10 20 30 50 60
```

Problem 3: Update Negative Values to Positive

Problem Statement:

Write a function to update all negative elements in a given array by converting them to positive.

cpp Copy Edit

```cpp
void convertNegativesToPositive(int arr[], int n);
```

Input:

First line: Integer n (number of elements)

Second line: n integers (can include negative values)

Output:

Print the updated array where all negative values are converted to positive.

Sample Input:

diff Copy Edit

```diff
6
-10 5 -3 8 0 -2
```

Sample Output:

Copy Edit

```
10 5 3 8 0 2
```

Problem 4: Find Second Largest Element

Problem Statement:

Write a function to find the second largest element in a given array of distinct integers.

```cpp
int findSecondLargest(int arr[], int n);
```

Input:

First line: Integer n (number of elements)

Second line: n distinct integers

Output:

Print the second largest element.

Sample Input:

```
5
25 10 60 45 30
```

Sample Output:

```
45
```

[1, 2, 60, 90, 10]

L1 → for ( ) $\longrightarrow$ max $\longrightarrow$ 90

(n)

(n)

L2 for ( ) < 90 $\longrightarrow$ 60

$\boxed{2n}$

$n + n \Rightarrow \cancel{2n} \Rightarrow O(n)$

Problem 5: Move All Zeros to End

Problem Statement:

Write a function to move all 0s in the array to the end while maintaining the order of other elements.

```cpp
void moveZerosToEnd(int arr[], int n);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Output:

Print the array after moving all 0s to the end.

Sample Input:

```
7
0 5 0 3 0 1 4
```

Sample Output:

```
5 3 1 4 0 0 0
```

Problem 6: Count Frequency of a Given Element

Problem Statement:

Write a function to count how many times a given number appears in the array.

```cpp
int countFrequency(int arr[], int n, int key);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Third line: Integer key to search for

Output:

Print the count of how many times the key appears.

Sample Input:

```
7
1 2 3 2 4 2 5
2
```

Sample Output:

```
3
```

Problem 7: Check if Array is Sorted

Problem Statement:

Write a function to check if the given array is sorted in non-decreasing order.

```cpp
bool isSorted(int arr[], int n);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Output:

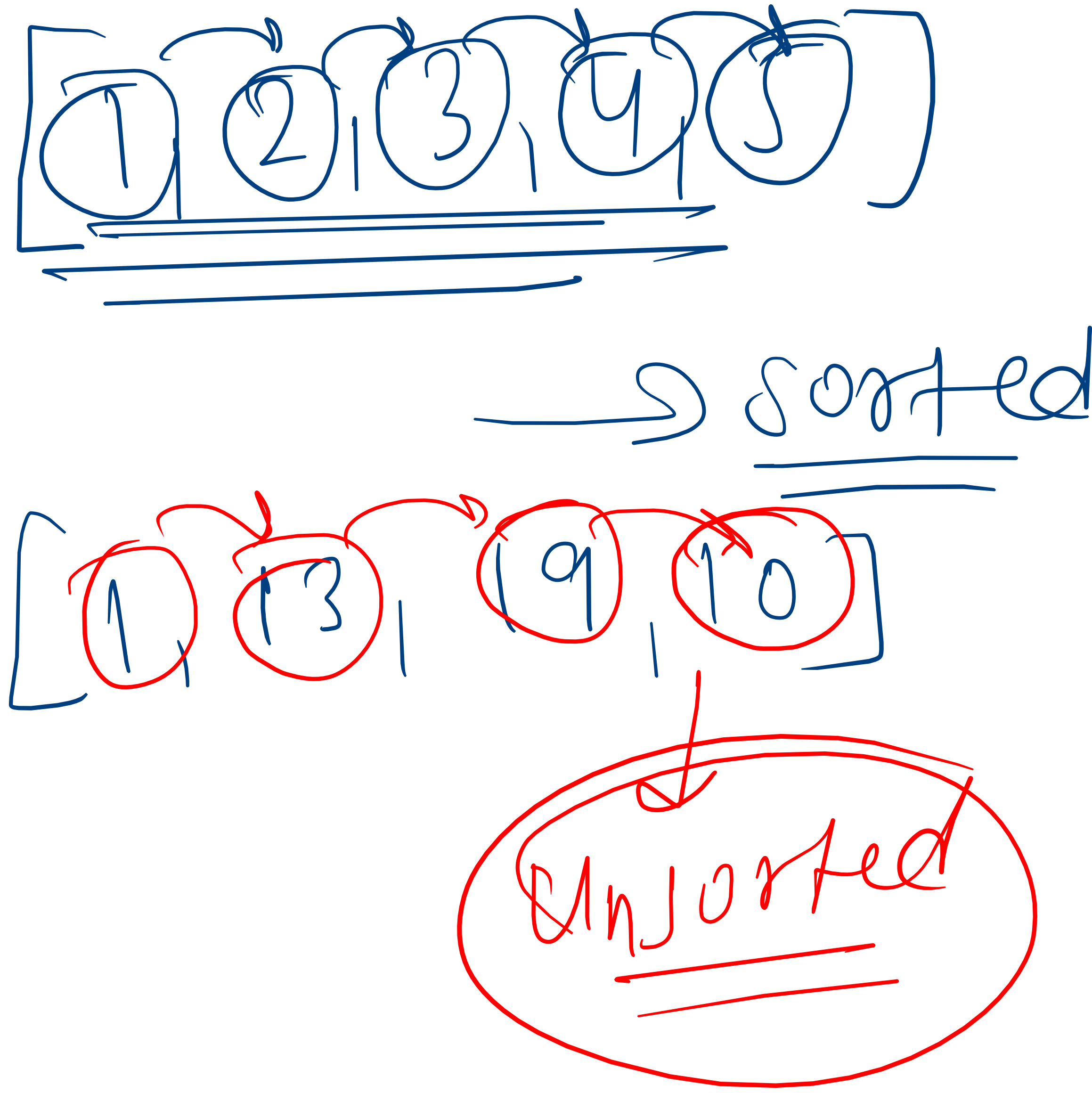
Print "YES" if the array is sorted, otherwise print "NO".

Sample Input:

```
5
1 2 3 4 5
```

Sample Output:

```objectivec
YES
```



Problem 8: Rotate Array to the Right by One

Problem Statement:

Write a function to rotate the array elements to the right by one position.

```cpp
void rotateRightByOne(int arr[], int n);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Output:

Print the array after rotating it right by one position.

Sample Input:

```
5
10 20 30 40 50
```

Sample Output:

```
50 10 20 30 40
```

Problem 9: Find First Repeating Element

Problem Statement:

Write a function to find and print the first repeating element in the array. If no element repeats, print -1.

```cpp
int findFirstRepeating(int arr[], int n);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Output:

Print the first element that appears more than once. If no element repeats, print -1.

Sample Input:

```
6
4 5 1 2 1 3
```

Sample Output:

```
1
```

Problem 10: Find Number of Positive, Negative, and Zero Elements

Problem Statement:

Write a function to count how many positive, negative, and zero elements are in the array.

```cpp
void countPosNegZero(int arr[], int n, int &pos, int &neg, int &zero);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Output:

Print three space-separated integers: count of positive, negative, and zero values.

Sample Input:

```diff
7
-3 0 2 -1 0 4 -2
```

Sample Output:

```
2 3 2
```

Problem 11: Frequency of Each Element

Problem Statement:

Write a function to find and print the frequency of **each unique element** in the given array.

Each element and its frequency should be printed only once.

```cpp
void printFrequencies(int arr[], int n);
```

Input:

First line: Integer `n` (number of elements)

Second line: `n` space-separated integers

Output:

Print each element and its frequency in the format:

`element: frequency`

Each element should be printed only once, in the order of their first appearance.

Sample Input:

```
7
1 3 5 3 2 3 1
```

Sample Output:

```makefile
1: 2
3: 3
5: 1
2: 1
```

Problem 12: Rotate Array by K Positions

Problem Statement:

Write a function to rotate the array to the right by `k` positions. The rotation should wrap around the array.

```cpp
void rotateRightByK(int arr[], int n, int k);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Third line: Integer k (number of positions to rotate)

Output:

Print the array after rotating it to the right by k positions.

Sample Input:

```
6
1 2 3 4 5 6
2
```

Sample Output:

```
5 6 1 2 3 4
```

Problem 13: Find Pair with Given Sum

Problem Statement:

Write a function to find any one pair of elements in the array whose sum is equal to a given target value.

```cpp
bool findPairWithSum(int arr[], int n, int target);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Third line: Integer target (sum to find)

Output:

If a pair exists, print the two numbers separated by space.

If no such pair exists, print -1.

Sample Input:

```
6
2 4 7 11 5 3
9
```

Sample Output:

```
4 5
```

Problem 14: Find Difference Between Max and Min

Problem Statement:

Write a function to find the difference between the maximum and minimum element in the array.

```cpp
int maxMinDifference(int arr[], int n);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Output:

Print the difference (maximum - minimum)

Sample Input:

```
5
10 20 5 8 30
```

Sample Output:

```
25
```

Problem 15: Find Unique Elements (Appear Only Once)

Problem Statement:

Write a function to print all elements that appear only once in the array.

```cpp
void printUniqueElements(int arr[], int n);
```

Input:

First line: Integer n (number of elements)

Second line: n integers

Output:

Print all unique elements (elements that appear only once), separated by space.

Order of output should be the same as original input.

Sample Input:

```
7
1 2 2 3 4 4 5
```

Sample Output:

```
1 3 5
```