1) You are given an array of n integers and an integer k. You need to rotate the array k times to the left or right based on a specified direction.

1. **Left Rotation**: Each element of the array should be shifted to its left neighbor's position, and the first element should move to the end of the array.
   Example: For an array [1, 2, 3, 4, 5] and k = 2 with left rotation, the resulting array will be [3, 4, 5, 1, 2].
2. **Right Rotation**: Each element of the array should be shifted to its right neighbor's position, and the last element should move to the beginning of the array.
   Example: For an array [1, 2, 3, 4, 5] and k = 2 with right rotation, the resulting array will be [4, 5, 1, 2, 3].

Task: Write a C program rotateArray that performs the rotation as specified.

2)You are given an array of integers. Implement a C program that performs the following operations:

1. **Count Occurrences**: Count how many times a given number appears in the array.
2. **Sum of Elements**: Calculate the sum of all elements in the array.
3. **Product of Elements**: Calculate the product of all elements in the array.

3)You are given an array of integers. Write a C program that computes and prints the square of each element in the array.

Sample Input1:
int arr[] = {1, 2, -3, 4, -5};

Sample Output1:
Squares of the array elements are: 1 4 9 16 25

Sample Input2:

int arr[] = {0, 3, -7, 8, 6};

Sample Output2:
Squares of the array elements are: 0 9 49 64 36

Sample Input3:
int arr[] = {-1, -2, -3, -4, -5};

Sample Output3:
Squares of the array elements are: 1 4 9 16 25

3)Write a C program to find the sum of the maximum and minimum elements of a given array. The program should:

- Accept the number of elements in the array.
- Accept the elements of the array from the user.
- Identify the maximum and minimum elements in the array.
- Calculate and output the sum of the maximum and minimum elements.

Sample input1:

Number of elements: 6

Elements of the array: 5, 12, 7, 9, 15, 3

Sample output1:

Sum of maximum and minimum elements: 18

Sample input2:

Number of elements: 1

Elements of the array: 7

Sample output2:

Sum of maximum and minimum elements: 14

Sample input3:

Number of elements: 4

Elements of the array: 4, 4, 4, 4

Sample output3:

Sum of maximum and minimum elements: 8

Sample input4:

Number of elements: 5

Elements of the array: -10, -3, -7, -1, -6

Sample output4:

Sum of maximum and minimum elements: -11

Sample input5:

Number of elements: 6

Elements of the array: -5, 3, 8, -2, 10, -7

Sample output5:

Sum of maximum and minimum elements: 3

4)squaring the duplicates

Write a C program that takes an array of integers as input and squares any duplicate elements in the array. The program should:

1. Accept the number of elements in the array.
2. Accept the elements of the array from the user.
3. Identify duplicate elements in the array.
4. Square the duplicate elements.
5. Output the modified array where only the duplicates have been squared.

Sample input1:

Enter the number of elements in the array: 6

Enter the elements of the array: -1 2 -1 3 2 3

Sample Output1:

Modified array: 1 4 9

Sample input2:

Enter the number of elements in the array: 5

Enter the elements of the array:1 2 3 4 5

Sample Output2:

Modified array:

Sample input3:

Enter the number of elements in the array: 4

Enter the elements of the array:7 7 7 7

Sample Output3:

Modified array: 49

Sample input4:

Enter the number of elements in the array: 7

Enter the elements of the array:5 6 7 8 5 6 7

Sample Output4:

Modified array: 25 36 49

5)Write a C program to find the difference between the maximum and minimum elements of a given array. The program should:

- Accept the number of elements in the array.
- Accept the elements of the array from the user.
- Calculate the maximum and minimum elements in the array.
- Output the difference between the maximum and minimum elements.

Sample input1:
Number of elements: 5
Elements of the array: 10, 3, 5, 8, 12
Sample output1:
Difference between maximum and minimum elements: 9

Sample input2:
Number of elements: 1
Elements of the array: 7
Sample output2:
Difference between maximum and minimum elements: 0

Sample input3:
Number of elements: 4
Elements of the array: 4, 4, 4, 4

Sample output3:
Difference between maximum and minimum elements: 0

Sample input4:
Number of elements: 6
Elements of the array: -5, 3, 8, -2, 10, -7
Sample output4:
Difference between maximum and minimum elements: 17

6) Write a C program to count how many elements in a given array are divisible by a specific number. The program should:

- Accept the number of elements in the array.
- Accept the elements of the array from the user.
- Accept the specific number (divisor) from the user.
- Count and output the number of elements in the array that are divisible by the given divisor.

Sample input1:
Enter the number of elements in the array: 6
Enter the elements of the array:15 30 22 44 55 60
Enter the divisor: 5
Sample output1:
Count of elements divisible by 5: 4

Sample input2:
Enter the number of elements in the array: 5
Enter the elements of the array:7 11 13 17 19
Enter the divisor: 2
Sample output2:
Count of elements divisible by 2: 0

Sample input3:
Enter the number of elements in the array: 4
Enter the elements of the array:12 24 36 48
Enter the divisor: 12
Sample output3:
Count of elements divisible by 12: 4

Sample input4:
Enter the number of elements in the array: 5
Enter the elements of the array:2 4 6 8 10
Enter the divisor: 1

Sample output4:
Count of elements divisible by 1: 5

Sample input5:
Enter the number of elements in the array: 6
Enter the elements of the array:-10 15 -20 25 30 -35
Enter the divisor: 5

Sample output5:
Count of elements divisible by 5: 6

7)Write a C program to calculate the sum of the elements in an array after removing all duplicate elements. The program should:

- Accept the number of elements in the array.
- Accept the elements of the array from the user.
- Remove any duplicate elements from the array.
- Calculate and output the sum of the remaining elements.

Sample input1:

Enter the number of elements in the array: 7
Enter the elements of the array:10 20 10 30 20 40 50

Sample output1:
Sum after removing duplicates: 150

Sample input2:
Enter the number of elements in the array: 5
Enter the elements of the array:5 10 15 20 25

Sample output2:
Sum after removing duplicates: 75

Sample input3:
Enter the number of elements in the array: 4
Enter the elements of the array:10 10 10 10

Sample output3:
Sum after removing duplicates: 10

Sample input4:
Enter the number of elements in the array: 6
Enter the elements of the array:-5 10 -5 20 10 -5

Sample output4:
Sum after removing duplicates: 25

Sample input5:
Enter the number of elements in the array: 1
Enter the elements of the array:42

Sample output5:
Sum after removing duplicates: 42

8)Write a C program to print all the negative elements in a given array. The program should:

- Accept the number of elements in the array.
- Accept the elements of the array from the user.
- Identify and print all the negative elements in the array.

Sample input1:
Enter the number of elements in the array: 8
Enter the elements of the array:12 -7 5 -3 10 -15 8 -2

Sample output1:
Negative elements in the array: -7, -3, -15, -2

Sample input2:
Enter the number of elements in the array: 5
Enter the elements of the array:1 2 3 4 5

Sample output2:
Negative elements in the array: None

Sample input3:
Enter the number of elements in the array: 4
Enter the elements of the array:-10 -20 -30 -40

Sample output3:
Negative elements in the array: -10, -20, -30, -40

Sample input4:
Enter the number of elements in the array: 6
Enter the elements of the array:0 -1 2 -3 4 -5

Sample output4:
Negative elements in the array: -1, -3, -5

Sample input5:
Enter the number of elements in the array: 1
Enter the elements of the array:-100

Sample output5:
Negative elements in the array: -100

9)Print the peak elements in array

Write a C program to find and print all the peak elements in a given array. A peak element is defined as an element that is greater than or equal to its neighbors. Specifically, for an element at index `i`, it is considered a peak if:

1. It is greater than or equal to both its left neighbor (`arr[i-1]`) and right neighbor (`arr[i+1]`), or
2. It is the first element in the array and greater than or equal to its right neighbor, or
3. It is the last element in the array and greater than or equal to its left neighbor.

The program should:

1. Accept the number of elements in the array.
2. Accept the elements of the array from the user.
3. Identify and print all the peak elements in the array.

Sample input1:
Enter the number of elements in the array: 7
Enter the elements of the array:1 3 20 4 1 0 6

Sample output1:
Peak elements in the array: 20,6

Sample input2:
Enter the number of elements in the array: 1

Enter the elements of the array: 7

Sample output2:
Peak elements in the array: 7

Sample input3:
Enter the number of elements in the array: 5
Enter the elements of the array:1 2 3 4 5

Sample output3:
Peak elements in the array: 5

Sample input4:
Enter the number of elements in the array: 6
Enter the elements of the array:10 9 8 7 6 5

Sample output4:
Peak elements in the array: 10, 5

Sample input5:
Enter the number of elements in the array: 8
Enter the elements of the array:1 3 2 5 4 8 6 7

Sample output5:
Peak elements in the array: 3, 5, 8, 7

10)Print the count of positive numbers

Write a C program to count and print the number of positive numbers in a given array. The program should:

1. Accept the number of elements in the array.

2. Accept the elements of the array from the user.
3. Count and print the number of positive numbers in the array.

Sample input1:
Enter the number of elements in the array: 6
Enter the elements of the array: -2 3 4 -1 0 6

Sample output1:
Count of positive numbers: 3

Sample input2:
Enter the number of elements in the array: 5
Enter the elements of the array: 0 0 0 0 0

Sample output2:
Count of positive numbers: 0

Sample input3:
Enter the number of elements in the array: 4
Enter the elements of the array: -1 -2 -3 -4

Sample output3:
Count of positive numbers: 0

11)sum of duplicates in an array

Write a C program to find and print the sum of all duplicate elements in a given array. The program should:

1. Accept the number of elements in the array.
2. Accept the elements of the array from the user.
3. Identify duplicate elements in the array.

4. Calculate and print the sum of these duplicate elements. Each duplicate element should be counted only once in the sum.

Sample input1:
Enter the number of elements in the array: 8
Enter the elements of the array: 4 5 6 7 4 8 6 4

Sample output1:
Sum of duplicate elements: 10

Sample input2:
Enter the number of elements in the array: 10
Enter the elements of the array: 2 3 5 7 3 5 8 9 5 3

Sample output2:
Sum of duplicate elements: 8

Sample input3:
Enter the number of elements in the array: 8
Enter the elements of the array: 7 8 9 7 8 7 10 10

Sample output3:
Sum of duplicate elements: 25

12)sum of even numbers in an array

Write a C program to find and print the sum of all even numbers in a given array. The program should:

1. Accept the number of elements in the array.
2. Accept the elements of the array from the user.
3. Calculate and print the sum of all even numbers in the array.

Sample input1:
Enter the number of elements in the array: 7
Enter the elements of the array:1 2 3 4 5 6 7

Sample output1:
Sum of even numbers: 12

Sample input2:
Enter the number of elements in the array: 5
Enter the elements of the array:2 4 6 8 10

Sample output2:
Sum of even numbers: 30

Sample input3:
Enter the number of elements in the array: 4
Enter the elements of the array:1 3 5 7

Sample output3:
Sum of even numbers: 0

Sample input4:
Enter the number of elements in the array: 6
Enter the elements of the array:10 15 20 25 30 35

Sample output4:
Sum of even numbers: 60

Sample input5:
Enter the number of elements in the array: 1
Enter the elements of the array:8

Sample output5:
Sum of even numbers: 8

13)find the non prime numbers in an array

Write a C program to find and print all non-prime numbers in a given array. A non-prime number is defined as any number that is less than or equal to 1, or a number that has divisors other than 1 and itself.

The program should:

1. Accept the number of elements in the array.
2. Accept the elements of the array from the user.
3. Identify and print all non-prime numbers in the array.

Sample input1:
```
Enter the number of elements in the array: 8
Enter the elements of the array:1 2 3 4 5 6 7 8
```

Sample output1:
```
Non-prime numbers in the array: 1, 4, 6, 8
```

Sample input2:
```
Enter the number of elements in the array: 5
Enter the elements of the array:2 3 5 7 11
```

Sample output2:
```
Non-prime numbers in the array: (none)
```

Sample input3:
```
Enter the number of elements in the array: 6
Enter the elements of the array:4 6 8 9 10 12
```

Sample output3:
```
Non-prime numbers in the array: 4, 6, 8, 9, 10, 12
```

14)Write a C program that takes a sequence of integers as input and replaces the peak number with the sum of its neighboring elements. A peak number is an element that is greater than both of its neighbors. If the sequence has no peak, the output should remain the same.

Input:

- The first line contains an integer n, the number of elements in the sequence.
- The second line contains n space-separated integers representing the sequence.

Output:

- Print the modified sequence where the peak element is replaced by the sum of its neighboring elements.

Sample input1:
4
1 2 4 3

Sample output1:
1 2 9 3

Explanation:
The peak is 4, which is greater than 2 and 3. It is replaced by 2+3=92 + 3 = 92+3=9.

Sample input2:
3
1 3 2

Sample output2:
1 3 2

Sample input3:

```
5
5 10 7 8 6
```

Sample output3:

```
5 22 7 8 6
```

Sample input4:

```
4
10 5 4 3
```

Sample output4:

```
10 5 4 3
```

Sample input5:

```
4
5 5 5 5
```

Sample output5:

```
5 5 5 5
```

Sample input6:

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

Sample output6:

```
1 2 3 4 5
```

Sample input7:

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

Sample output7:

```
1 5 2 7 3 12  4
```

15)Write a C program that takes an array of integers as input and prints only the positive numbers from the array in the order they appear.

Input:

- The first line contains an integer n, the number of elements in the array.
- The second line contains n space-separated integers representing the elements of the array.

Output:

- Print the positive numbers from the array in a single line, separated by spaces.

Sample input1:
6
1 3 -4 -5 2 3

Sample output1:
1 3 2 3

Sample input2:
5
-1 -2 -3 -4 -5

Sample output2:

Sample input3:
4
0 2 -2 3

Sample output3:
2 3

16)Write a C program that takes an array of integers as input and prints the sum of the first element, second element, last element, and second-to-last element of the array. If the array has fewer than four elements, sum the available elements accordingly.

Input:

- The first line contains an integer n, the number of elements in the array.
- The second line contains n space-separated integers representing the elements of the array.

Sample input1:
5
1 2 4 5 5

Sample output1:
13

Explanation:
The sum is calculated as 1 + 2 + 5 + 5 = 13.

Sample input2:
3
1 2 3

Sample output2:
4

Sample input3:
1
5

Sample output3:
5

17)Write a C program that takes an array of integers as input, sorts the array in non-decreasing order, and then prints the median of the array. The median is the middle element of a sorted array if the number of elements is odd, or the average of the two middle elements if the number of elements is even.

Input:

- The first line contains an integer `n`, the number of elements in the array.
- The second line contains `n` space-separated integers representing the elements of the array.

Output:

- Print the median of the sorted array.

Sample input1:
5
3 1 4 1 5

Sample output1:
3

Sample input2:
4
7 1 3 5

Sample output2:
4

Sample input3:
1
42

Sample output3:
42

Sample input4:
5
-10 20 5 -30 15

Sample output4:
5

Sample input5:
4
7 7 7 7

Sample output5:
7

18)Write a C program that takes an array of integers as input and prints the average of the elements in the array. The average should be a floating-point number.

Input:

- The first line contains an integer `n`, the number of elements in the array.
- The second line contains `n` space-separated integers representing the elements of the array.

Output:

- Print the average of the array elements as a floating-point number.

Sample input1:
5
1 2 3 4 5
Sample output1:
3.00

Sample input2:
4
10 20 30 40

Sample output2:
25.00

Sample input3:
1
10
Sample output3:
10.00

Sample input4:
3
5 5 5
Sample output4:
5.00

Sample input5:
4
0 0 0 0
Sample output5:
0.00

19)Write a C program that takes an array of integers as input and prints the two elements whose sum is closest to zero. The program should first check that the number of elements in the array is between 2 and 10. If the number of elements exceeds this range, print "Invalid".

Input:

- The first line contains an integer `n`, the number of elements in the array.
- The second line contains `n` space-separated integers representing the elements of the array.

Output:

- If n is not within the range 2 to 10, print "Invalid".
- Otherwise, print the two elements whose sum is closest to zero, in the order they appear in the array.

Sample input1:
4
-1 -10 8 2
Sample output1:
-1 2

Sample input2:
5
-2 1 -3 7 5
Sample output2:
-2 1

Sample input3:
1
-5
Sample output3:
Invalid

Sample input4:
3
-7 -5 -3
Sample output4:
-5 -3

Sample input5:
11
1 2 3 4 5 6 7 8 9 10 11
Sample output5:
Invalid

20)Write a C program that takes an array of integers as input and prints the unique elements in the array. An element is considered unique if it appears exactly once in the array. The output should list all unique elements in the order they appear in the array.

Input:

- The first line contains an integer `n`, the number of elements in the array.
- The second line contains `n` space-separated integers representing the elements of the array.

Output:

- Print the unique elements in the array, separated by spaces, in the order they first appear. If there are no unique elements, print "No unique elements".

Sample input1:
6
0 5 3 5 2 3
Sample output1:
0 2

Sample input2:
7
1 2 2 3 3 4 5
Sample output2:
1 4 5

Sample input3:
4
10 20 30 40
Sample output3:
10 20 30 40

Sample input4:
5
7 7 7 7 7
Sample output4:
No unique elements

Sample input5:
0
Sample output5:
No unique elements

21)Write a C program that takes an array of integers as input and prints the second largest element in the array

Input:

- The first line contains an integer n, the number of elements in the array.
- The second line contains n space-separated integers representing the elements of the array.

Output:

- Print the second largest element in the array. If there are fewer than two distinct elements, print "Not enough distinct elements".

Sample input1:
6
10 20 4 20 30 30
Sample output1:
20

Sample input2:
5
5 5 5 5 5
Sample output2:

Not enough distinct elements

Sample input3:
4
7 3 9 5
Sample output3:
7

Sample input4:
6
1 2 2 1 2 1
Sample output4:
1

Sample input5:
7
-5 1 -3 1 2 -3 2
Sample output5:
1

22)Write a C program that takes an array of integers as input and prints the count of elements that are repeated (i.e., appear more than once).

Input:

- The first line contains an integer `n`, the number of elements in the array.
- The second line contains `n` space-separated integers representing the elements of the array.

Output:

- Print the count of elements that are repeated in the array.

Sample input1:

6
1 2 2 3 4 4
Sample output1:
2

Sample input2:
5
1 1 1 2 2
Sample output2:
2

Sample input3:
4
1 2 3 4
Sample output3:
0

Sample input4:
6
5 5 5 5 5 5
Sample output4:
1

Sample input5:
6
8 8 9 9 9 9
Sample output5:
2

23)Write a C program that takes an array of integers as input and prints the even numbers first, followed by the odd numbers. The numbers should maintain their original relative order within their respective even or odd groups.

Input:

- The first line contains an integer n, the number of elements in the array.
- The second line contains n space-separated integers representing the elements of the array.

Output:

- Print all even numbers in the order they appear, followed by all odd numbers in the order they appear.

Sample input1:
7
4 5 2 7 8 3 6
Sample output1:
4 2 8 6 5 7 3

Sample input2:
5
1 3 5 2 4
Sample output2:
2 4 1 3 5

Sample input3:
4
2 4 6 8
Sample output3:
2 4 6 8

24)Write a C program that takes an array of integers as input and prints the array after removing all even numbers. The remaining numbers (odd numbers) should maintain their original order.

Input:

- The first line contains an integer `n`, the number of elements in the array.
- The second line contains `n` space-separated integers representing the elements of the array.

Output:

- Print the elements of the array after removing all even numbers. If there are no odd numbers, print "No odd numbers".

Sample input1:
7
4 5 2 7 8 3 6
Sample output1:
5 7 3

Sample input2:
5
1 2 3 4 5
Sample output2:
1 3 5

Sample input3:
4
2 4 6 8
Sample output3:
No odd numbers

25)Write a C program that takes an array of integers and a target integer as input and prints the array after removing the first occurrence of the target integer. The remaining elements should maintain their original order.

Input:

- The first line contains an integer `n`, the number of elements in the array.
- The second line contains `n` space-separated integers representing the elements of the array.
- The third line contains the target integer to be removed.

Output:

- Print the elements of the array after removing the first occurrence of the target integer. If the target integer is not found in the array, print "Element not found".

Sample input1:
```
7
1 2 3 4 5 3 6
3
```
Sample output1:
```
1 2 4 5 3 6
```

Sample input2:
```
5
10 20 30 40 50
25
```
Sample output2:
```
Element not found
```

Sample input3:
```
1
5
5
```

Sample output3:

Sample input4:
1
8
5
Sample output4:
8

Sample input5:
5
1 2 3 4 5
5
Sample output5:
1 2 3 4

26)Write a C program that takes an array of integers and a chunk size as input and prints the array divided into chunks of the specified size. If the array cannot be evenly divided into chunks of the given size, the last chunk should contain the remaining elements.

Input:

- The first line contains an integer `n`, the number of elements in the array.
- The second line contains `n` space-separated integers representing the elements of the array.
- The third line contains an integer `chunk_size`, specifying the size of each chunk.

Output:

- Print the array divided into chunks of the specified size. Each chunk should be enclosed in square brackets and separated by spaces. If the array cannot be evenly divided, the last chunk should contain the remaining elements.

Sample input1:

```
8
1 2 3 4 5 6 7 8
2
```

Sample output1:

```
[1 2] [3 4] [5 6] [7 8]
```

Sample input2:

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

Sample output2:

```
[1 2 3] [4]
```

Sample input3:

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

Sample output3:

```
[10 20 30 40 50]
```

Sample input4:

```
3
7 8 9
5
```

Sample output4:

```
[7 8 9]
```

Sample input5:

```
10
1 2 3 4 5 6 7 8 9 10
4
```

Sample output5:

```
[1 2 3 4] [5 6 7 8] [9 10]
```

27)Write a C program to reverse an array of integers. The program should:

1. Take the number of elements in the array as input.
2. Take the array elements as input.
3. Reverse the array.
4. Print the reversed array.

Sample input1:

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

Sample output1:

```
5 4 3 2 1
```