First Half Ascending, Second Half Descending:

```
def printOrder(arr, n):
    arr.sort()
    for i in range(n // 2):
        print(arr[i], end = " ")
    for j in range(n - 1, n // 2 -1, -1) :
        print(arr[j], end = " ")

arr = [ 5, 4, 6, 2, 1, 3, 8, -1 ]
n = len(arr)
printOrder(arr, n)
```

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

Frequency of elements in Array:

```
def find_frequency(arr):
    frequency_dict = {}

    for element in arr:
        if element in frequency_dict:
            frequency_dict[element] += 1
        else:
            frequency_dict[element] = 1

    return frequency_dict

array = [1, 2, 3, 2, 4, 1, 5, 1, 3, 3, 2]
frequency = find_frequency(array)
print(frequency)
```

```
{1: 3, 2: 3, 3: 3, 4: 1, 5: 1}
```

Distinct Element:

```
def count_distinct_elements(arr):
    distinct_elements = set(arr)
    count = len(distinct_elements)
    return count

# Example usage
array = [1, 2, 3, 2, 4, 1, 5, 1, 3, 3, 2]
distinct_count = count_distinct_elements(array)
print(distinct_count)
```

Output = 5

Subset of another Array:

```
def isSubset(arr1, arr2, m, n):
    i = 0
    j = 0
    for i in range(n):
        for j in range(m):
            if(arr2[i] == arr1[j]):
                break
        if (j == m):
            return 0
    return 1
arr1 = [11, 12, 13, 21, 30, 70]
arr2 = [11, 30, 70, 12]

m = len(arr1)
n = len(arr2)

if(isSubset(arr1, arr2, m, n)):
    print("arr2[] is subset of arr1[] ")
else:
    print("arr2[] is not a subset of arr1[]")
```

```
arr2[] is subset of arr1[]
```

Monotonic Array:

main.py

```
def is_monotonic(nums):
    n = len(nums)
    increasing = decreasing = True

    for i in range(1, n):
        if nums[i] < nums[i-1]:
            increasing = False
        if nums[i] > nums[i-1]:
            decreasing = False

    return increasing or decreasing

nums = [1, 2, 2, 3]
is_monotonic_array = is_monotonic(nums)
print(is_monotonic_array)
```

```
True
```

Roman to Integer:

main.py

```
def roman_to_integer(roman):
    roman_values = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
    total = 0
    prev_value = 0
    for symbol in roman[::-1]:
        value = roman_values[symbol]
        if value < prev_value:
            total -= value
        else:
            total += value
        prev_value = value
    return total
roman_numeral = 'XIV'
integer_value = roman_to_integer(roman_numeral)
print(integer_value)
```

```
14
```

XORed Array:

main.py

```python
def decode(encoded, first):
    arr = [first]
    for num in encoded:
        decoded_num = arr[-1] ^ num
        arr.append(decoded_num)
    return arr
encoded = [1, 2, 3]
first = 1
original_arr = decode(encoded, first)
print(original_arr)
```

```
[1, 0, 2, 1]
```

Other Array Programs:

[Reverse the array]

[Find the maximum and minimum element in an array]

[Find the "Kth" max and min element of an array]

[Given an array which consists of only 0, 1 and 2. Sort the array without using any sorting algo]

[Move all the negative elements to one side of the array]

[Find the Union and Intersection of the two sorted arrays.]

[Write a program to cyclically rotate an array by one.]

[find Largest sum contiguous Subarray [V. IMP]]

[Minimise the maximum difference between heights [V.IMP]]

[Minimum no. of Jumps to reach end of an array]

[find duplicate in an array of N+1 Integers]

[Merge 2 sorted arrays without using Extra space.]

Kadane's Algo [V.V.V.V.V IMP]

Merge Intervals

Next Permutation

Count Inversion

Best time to buy and Sell stock

find all pairs on integer array whose sum is equal to given number

find common elements In 3 sorted arrays

Rearrange the array in alternating positive and negative items with O(1) extra space

Find if there is any subarray with sum equal to 0

Find factorial of a large number

find maximum product subarray

Find longest coinsecutive subsequence

Given an array of size n and a number k, fin all elements that appear more than " n/k " times.

Maximum profit by buying and selling a share atmost twice

Find whether an array is a subset of another array

Find the triplet that sum to a given value

Trapping Rain water problem

Chocolate Distribution problem

Smallest Subarray with sum greater than a given value

Three way partitioning of an array around a given value

Minimum swaps required bring elements less equal K together

Minimum no. of operations required to make an array palindrome

Median of 2 sorted arrays of equal size

Median of 2 sorted arrays of different size