Algorithm Questions (You can solve these questions with any programming language or pseudo-code)

1.Divide all elements of the array by the value of the largest element of this array

```
from math import gcd as __gcd

# Function to return the largest number
# that divides the maximum elements
# from the given array
def findLargest(arr, n):

    # Finding gcd of all the numbers
    # in the array
    gcd = 0
    for i in range(n):
        gcd = __gcd(arr[i], gcd)
    return gcd

# Driver code
if __name__ == '__main__':
    arr = [3, 6, 9]
    n = len(arr)

    print(findLargest(arr, n))
```

2.Find the number and value of the first positive element in the array.

posIndex = -1;

```
for (i = 0; i < n; i++){
     if (a[i] > 0) {
          posIndex = i;
          break;
     }
}
```

3.Given an array containing positive and negative numbers. Replace all elements of the array with opposite signs.

For example, given the array [1, -5, 0, 3, -4]. After conversion, you should get [-1, 5, 0, -3, 4].

```
# list integers to negative and vice-versa
def Convert(lst):
    return [ -i for i in lst ]

# Driver code
lst = [-1, 2, 3, -4, 5, -6, -7]
print(Convert(lst))
```

Output:

```
[1, -2, -3, 4, -5, 6, 7]
```

4. Find the minimum and maximum elements in the array, and swap them.

5. Find the minimum and maximum elements in a one-dimensional array. Calculate their difference.

Pseudo-Code

```
int[] getMinMax(int A[], int n)
{
    int max = A[0]
    int min = A[0]
    for ( i = 1 to n-1 )
    {
        if ( A[i] > max )
            max = A[i]
        else if ( A[i] < min )
            min = A[i]
    }
    // By convention, let ans[0] = maximum and ans[1] = minimum
    int ans[2] = {max, min}
    return ans
}
```

```
def swap_min_max(l):

    max_value = max(l)
    min_value = min(l)

    # get all potential indices
    idx_min = [i for i,e in enumerate(l) if e==min_value]
    idx_max = [i for i,e in enumerate(l) if e==max_value]

    # replace values
    for idx in idx_min:
        l[idx] = max_value
    for idx in idx_max:
        l[idx] = min_value

    return l

>>>> swap_min_max([4, 5, 5, 2, 1, 1])
[4, 1, 1, 2, 5, 5]

>>>> swap_min_max([3, 4, 5, 2, 1])
[3, 4, 1, 2, 5]
```

6.Find the sum of those array elements that have both even and negative values at the same time

```
class Sumofnumbers:
    # find sum of numbers
    # according to categories
    def Sum(self, list):

        # counter for sum
        # of negative numbers
        neg_sum = 0

        # counter for sum of
        # positive even numbers
        pos_even_sum = 0

        # counter for sum of
        # positive odd numbers
        pos_odd_sum = 0

        for num in list:

            # converting number
            # to integer explicitly
            num = int(num)
            # if negative number
            if(num < 0):

                # simply add
                # to the negative sum
                neg_sum += num
            # if positive number
            else:

                # if even positive
                if(num % 2 == 0):

                    # add to positive even sum
                    pos_even_sum += num
                else:

                    # add to positive odd sum
                    pos_odd_sum += num
        print("Sum of negative numbers is ",
              neg_sum)
        print("Sum of even positive numbers is ",
              pos_even_sum)
        print("Sum of odd positive numbers is ",
              pos_odd_sum)
# input a list of numbers
list_num = [1, -1, 50, -2, 0, -3]

# creating an object of class
obj = Sumofnumbers()

# calling method for
# largest sum of all numbers
obj.Sum(list_num)
```

7.Find the minimum value in the array among the elements with odd indices.

```
z= [1,8,-4,-9]

def min_odd(x):
    for i in x:
        if (i%2!=0):
    return min(i)

y = min_odd(z)
print (y)
```

8.Given a one-dimensional array. Write on a screen only those array elements that are greater than the arithmetic average of all array elements

```
def printAboveAvg(arr, a):

    # Find average
    avg = 0
    for i in range(a):
        avg = avg + arr[i]

    avg = avg // a

    # Print elements greater than
    # average
    for i in range(a):
        if arr[i] > avg:
            print(arr[i], end = " ")

# Driver Program
arr = [5, 4, 6, 9, 10]
a = len(arr)
printAboveAvg(arr, a)
```

Output:

```
9 10
```

9.Find the sum of the positive elements of an array.

```
>>> a = [1, 2, 3, -4, 5, -3, 7, 8, 9, 6, 4, -7]
>>> sum(x for x in a if x > 0)
45
```

11.Replace the minimum element in a one-dimensional array with the arithmetic average of all elements in the array.

```
def findAverage(arr, N, K):
    start = None
    end = None
    for i in range(N):
        sum = 0

        # Start limit is max(i-K, 0)
        start = max(i - K, 0)

        # End limit in min(i+K, N-1)
        end = min(i + K, N - 1)
        cnt = end - start
        for j in range(start, end + 1):

            # Skipping the current element
            if j == i:
                continue
            sum += arr[j]
        print((sum // cnt), end= " ")

# Driver Code
arr = [9, 7, 3, 9, 1, 8, 11]
N = len(arr)
K = 2
findAverage(arr, N, K)
```

12.In the array, determine the indices of elements whose value is not less than X, but not more than Y. The values of X and Y are set from the keyboard.

```
def findElement(arr, n):
    # leftMax[i] stores maximum of arr[0..i-1]
    leftMax = [None] * n
    leftMax[0] = arr[0]

    # Fill leftMax[]1..n-1]
    for i in range(1, n):
        leftMax[i] = max(leftMax[i-1], arr[i-1])

    # Initialize minimum from right
    rightMin = [None]*n
    rightMin[n-1] = arr[n-1]

    # Fill rightMin
    for i in range(n-2, -1, -1):
        rightMin[i] = min(rightMin[i+1], arr[i])
    # Traverse array from right
    for i in range(1, n-1):

        # Check if we found a required element
        # for ith element, it should be more than maximum of array
        # elements [0....i-1] and should be less than the minimum of
        # [i+1.....n-1] array elements
        if leftMax[i-1] <= arr[i] and arr[i] <= rightMin[i+1]:
            return i
    # If there was no element matching criteria
    return -1

# Driver program
if __name__ == "__main__":

    arr = [5, 1, 4, 3, 6, 8, 10, 7, 9]
    n = len(arr)
    print("Index of the element is", findElement(arr, n))
```

13.In an array consisting of N fractional elements, find the maximum absolute value of the element of the array.

```
arr = [25, 11, 7, 75, 56];

#Initialize max with first element of array.
max = arr[0];

#Loop through the array
for i in range(0, len(arr)):
   #Compare elements of array with max
  if(arr[i] > max):
     max = arr[i];
print("Largest element present in given array: " + str(max));
```

14Get the arithmetic average of all the even elements of the array standing in odd places. I remind you that the very first element has the ordinal number 0.

```
def arrangeOddAndEven(arr,  n):
    oddInd = 1
    evenInd = 0
    while (True):

        while (evenInd < n and arr[evenInd] % 2 == 0):
            evenInd += 2

        while (oddInd < n and arr[oddInd] % 2 == 1):
            oddInd += 2

        if (evenInd < n and oddInd < n):
                temp = arr[evenInd]
                arr[evenInd] = arr[oddInd]
                arr[oddInd] = temp;

        else:
            break

# function to print the array
def printArray(arr,  n):
    for i in range(0,n):
        print(arr[i] , "",end="")

# Driver function
def main():
    arr = [ 3, 6, 12, 1, 5, 8 ]
    n = 6
    print("Original Array: ",end="")
    printArray(arr, n)
    arrangeOddAndEven(arr, n)
    print("\nModified Array: ",end="")
    printArray(arr, n)
```

```
if __name__ == '__main__':
    main()
```

15.Which sum of the array elements greater – from the first to the element with the number K or from the element with the number K+1 to the last. The K value is entered from the keyboard.

```
def findK(arr, size, N):

    # Sorting the array in increasing order
    arr = sorted(arr)
    temp_sum = 0

    # Loop through all the elements of the array
    for i in range(size):
        temp_sum += arr[i]

        # Checking if sum of array equals N
        if (N - temp_sum == arr[i] * (size - i - 1)):
            return arr[i]
    return -1

# Driver code
arr = [3, 1, 10, 4, 8]
size = len(arr)
N = 16

print(findK(arr, size, N))
```

16.Find the sum and product of the elements of a one-dimensional numeric array.

```
numbers = [1, 2, 3]
total = sum(numbers)

product = 1
for i in numbers:
    product *= i
```

17.Find the sum and the number of array elements that are in the interval of [a; b]. The interval boundaries are entered from the keyboard.

```
>>> sum(range(102, 2001, 3))
664650
```

To make it into a robust function:

```
def sum_range_divisible(start, end, divisor):
    while start % divisor != 0:
        start += 1
    return sum(range(start, end, divisor))
```

18.In the given array of numbers, count the number of positive and the number of negative elements.

```
list1 = [10, -21, 4, -45, 66, -93, 1]
pos_count, neg_count = 0, 0
# iterating each number in list
for num in list1:
    # checking condition
    if num >= 0:
        pos_count += 1
    else:
        neg_count += 1

print("Positive numbers in the list: ", pos_count)
print("Negative numbers in the list: ", neg_count)
```

19.In a one-dimensional array, find the maximum of the negative elements and swap it with the last element of the array.

```
# Stores the resultant value
    # of K
    res = 0
    # Sort the array arr[]
    arr = sorted(arr)
    # Initialize two variables to
    # use two pointers technique
    l = 0
    r = len(arr) - 1
    # Iterate until the value of
    # l is less than r
    while (l < r):
        # Find the value of the sum
        sum = arr[l] + arr[r]
        # If the sum is 0, then the
        # resultant element is found
        if (sum == 0):
            res = max(res, max(arr[l], arr[r]))
            return res
        # If the sum is negative
        elif (sum < 0):
            l += 1
        # Otherwise, decrement r
        else:
            r -=1
    return res
# Driver Code
if __name__ == '__main__':
    arr =[3, 2, -2, 5, -3]
    print(largestNum(arr))
```

20.In an array consisting of positive and negative numbers, determine how many elements in absolute value are greater than the maximum element.

```
def maxCount(a):
    # Counting frequencies of elements
    freq = {}
    for i in range(n):
        if (a[i] in freq):
            freq[a[i]] += 1
        else:
            freq[a[i]] = 1
    # Finding max sum of adjacent indices
    ans = 0
    for key, value in freq.items():
        if (key+1 in freq) :
            ans = max(ans, freq[key] + freq[key + 1])
    return ans
# Driver Code
n = 5
arr = [2, 2, 3, 4, 5]
print(maxCount(arr))
```

21.You need to determine the number of elements whose value is greater than the neighbor elements of the array.

```
def printElements(arr, n):
    # Traverse array from index 1 to n-2
    # and check for the given condition
    for i in range(1, n - 1, 1):
        if (arr[i] > arr[i - 1] and
            arr[i] > arr[i + 1]):
            print(arr[i], end = " ")
# Driver Code
if __name__ == '__main__':
    arr = [2, 3, 1, 5, 4, 9, 8, 7, 5]
    n = len(arr)
    printElements(arr, n)
```

22.Let's say there is a one-dimensional array containing numbers from 0 to 49 inclusive. It is required to replace all its elements whose values are less than 15, by -1.

```
def ReplaceElements(arr, n):
    # Nothing to do when array size is 1
    if (n <= 1):
        return
    # store current value of arr[0]
    # and update it
    prev = arr[0]
    arr[0] = arr[0] + arr[1]
    # Update rest of the array elements
    for i in range(1, n - 1):
        # Store current value of
        # next iteration
        curr = arr[i]
        # Update current value using
        # previews value
        arr[i] = prev + arr[i + 1]
        # Update previous value
        prev = curr
    # Update last array element separately
    arr[n - 1] = prev + arr[n - 1]
# Driver Code
if __name__ == "__main__":
    arr = [ 2, 3, 4, 5, 6 ]
    n = len(arr)
    ReplaceElements(arr, n)
    # Print the modified array
    for i in range(n):
        print (arr[i], end = " ")
```

23.Fill the array with randomly generated numbers in the interval of [a; b].

```
import numpy as np
def Rand(start, end, num):
    res = []
    for j in range(num):
        res.append(np.random.randint(start, end))
    return res
# Driver Code
num = 10
start = 20
end = 40
print(Rand(start, end, num))
```

24.Find the sum of the indices of the minimum and maximum elements of the array.

```
def getMin(arr, n):
   res = arr[0]
   for i in range(1, n):
      res = min(res, arr[i])
   return res
```

```
# Function to find maximum element
def getMax(arr, n):
    res = arr[0]
    for i in range(1, n):
        res = max(res, arr[i])
    return res
# Function to get Sum
def findSum(arr, n):
    min = getMin(arr, n)
    max = getMax(arr, n)
    return min + max
# Function to get product
def findProduct(arr, n):
    min = getMin(arr, n)
    max = getMax(arr, n)
    return min * max
# Driver Code
if __name__ == "__main__":
    arr = [ 12, 1234, 45, 67, 1 ]
    n = len(arr)
    # Sum of min and max element
    print("Sum = " , findSum(arr, n))
    # Product of min and max element
    print("Product = " , findProduct(arr, n))
```

25.Find the sum of the array elements that lie between two zeros. It is guaranteed that the array will contain exactly two zeros and they will not be in neighbor positions.

```
def minAbsSumPair(arr,arr_size):
    inv_count = 0
    # Array should have at least
    # two elements
    if arr_size < 2:
        print("Invalid Input")
        return
    # Initialization of values
    min_l = 0
    min_r = 1
    min_sum = arr[0] + arr[1]
    for l in range (0, arr_size - 1):
        for r in range (l + 1, arr_size):
            sum = arr[l] + arr[r]
            if abs(min_sum) > abs(sum):
                min_sum = sum
                min_l = l
                min_r = r
    print("The two elements whose sum is minimum are",
            arr[min_l], "and ", arr[min_r])
# Driver program to test above function
arr = [1, 60, -10, 70, -80, 85]
minAbsSumPair(arr, 6);
```

26.Replace the values of all even-numbered array elements with 999.

```
import numpy as np
a = np.array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
odd_values = (a%2 == 1)
a[odd_values] = -1
# array([-1,  2, -1,  4, -1,  6, -1,  8, -1, 10])

# or you can do it in one shot:
a[a%2 == 1] = -1
```

27.Rearrange the elements of the given array in reverse order, that is, reverse the array.

```
#The original array
arr = [11, 22, 33, 44, 55]
print("Before reversal Array is :",arr)
arr.reverse() #reversing using reverse()
print("After reversing Array:",arr)
```

Output:

```
Before reversal Array is : [11, 22, 33, 44, 55]
After reversing Array: [55, 44, 33, 22, 11]
```

28.Calculate the sum of the absolute values of the array elements located after the first negative element. It is guaranteed that there will be at least one negative element in the array.

```
def counterlist(lst):
    it = iter(lst)
    for i in it:
        if i < 0:
            print('Reqemlerin cemi:', sum(it))
```

29.Calculate the product of the array elements located before the first negative element. It is guaranteed that there will be a single negative element in the array that will not be in the 0 place.

```
def multiplyList(myList) :
    # Multiply elements one by one
    result = 1
    for x in myList:
         result = result * x
    return result
# Driver code
list1 = [1, 2, 3]
print(multiplyList(list1))
```

30.In a one-dimensional array, find the sum of the elements that are between the minimum and maximum elements. The minimum and maximum elements themselves are not included in the sum.

```
def miniMaxSum(arr):
  arr_sorted = sorted(arr)
  return sum(arr_sorted[:4]), sum(arr_sorted[-4:])

print(miniMaxSum([1,2,3,4,5]))
print(miniMaxSum([5,5,5,5,5]))
```

Output:

```
>>> print(miniMaxSum([1,2,3,4,5]))
(10, 14)
>>> print(miniMaxSum([5,5,5,5,5]))
(20, 20)
```