Find minimum and maximum element in an array

Basic Accuracy: 59.4% Submissions: 42639 Points: 1

Given an array **A** of size **N** of integers. Your task is to find the **minimum and maximum** elements in the array.

Example 1:

```
Input:
N = 6
A[] = {3, 2, 1, 56, 10000, 167}
Output:
min = 1, max =  10000
```

Example 2:

```
Input:
N = 5
A[]  = {1, 345, 234, 21, 56789}
Output:
min = 1, max = 56789
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **getMinMax()** which takes the array **A[]** and its size **N** as inputs and returns the **minimum and maximum** element of the array.

Reverse a String

Basic Accuracy: 56.22% Submissions: 76778 Points: 1

You are given a string s. You need to reverse the string.

Example 1:

```
Input:
s = Geeks
Output: skeeG
```

Example 2:

```
Input:
s = for
Output: rof
```

Your Task:

You only need to complete the function **reverseWord()** that takes s as parameter and returns the reversed string.

Expected Time Complexity: O(|S|).
Expected Auxiliary Space: O(1).

Constraints:
1 <= |s| <= 10000

Sort The Array

Basic Accuracy: 75.41% Submissions: 14749 Points: 1

Given a random set of numbers, Print them in sorted order.

Example 1:

```
Input:
N = 4
arr[] = {1, 5, 3, 2}
Output: {1, 2, 3, 5}
Explanation: After sorting array will
be like {1, 2, 3, 5}.
```

Example 2:

```
Input:
N = 2
arr[] = {3, 1}
Output: {1, 3}
Explanation: After sorting array will
be like {3, 1}.
```

Your Task:

You don't need to read input or print anything. Your task is to complete the function **sortArr()** which takes the list of integers and the size N as inputs and returns the modified list.

Kth smallest element

Medium Accuracy: 46.66% Submissions: 100k+ Points: 4

Given an array **arr[]** and an integer **K** where K is smaller than size of array, the task is to find the **K^{th} smallest** element in the given array. It is given that all array elements are distinct.

Example 1:

```
Input:
N = 6
arr[] = 7 10 4 3 20 15
K = 3
Output : 7
Explanation :
3rd smallest element in the given
array is 7.
```

Example 2:

```
Input:
N = 5
arr[] = 7 10 4 20 15
K = 4
Output : 15
Explanation :
4th smallest element in the given
array is 15.
```

Your Task:
You don't have to read input or print anything. Your task is to complete the function **kthSmallest()** which takes the array **arr[]**, integers **l** and **r** denoting the **starting** and **ending** index of the array and an integer **K** as input and returns the **K^{th}** smallest element.

Find the Frequency

Easy Accuracy: 59.34% Submissions: 23724 Points: 2

Given a vector of **N** positive integers and an integer **X**. The task is to find the **frequency** of X in vector.

Example 1:

```
Input:
N = 5
vector = {1, 1, 1, 1, 1}
X = 1
Output:
5
Explanation: Frequency of 1 is 5.
```

Your Task:
Your task is to complete the function **findFrequency**() which should count the frequency of X and return it.

Sort an array of 0s, 1s and 2s

Easy Accuracy: 51.36% Submissions: 100k+ Points: 2

Given an array of size N containing only 0s, 1s, and 2s; sort the array in ascending order.

Example 1:

```
Input:
N = 5
arr[]= {0 2 1 2 0}
Output:
0 0 1 2 2
Explanation:
0s 1s and 2s are segregated
into ascending order.
```

Example 2:

```
Input:
N = 3
arr[] = {0 1 0}
Output:
0 0 1
Explanation:
0s 1s and 2s are segregated
into ascending order.
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **sort012()** that takes an array arr and N as input parameters and **sorts the array in-place.**

Subarray with given sum

Easy Accuracy: 39.71% Submissions: 100k+ Points: 2

Given an unsorted array **A** of size **N** that contains only non-negative integers, find a continuous sub-array which adds to a given number **S**.

Example 1:

```
Input:
N = 5, S = 12
A[] = {1,2,3,7,5}
Output: 2 4
Explanation: The sum of elements
from 2nd position to 4th position
is 12.
```

Example 2:

```
Input:
N = 10, S = 15
A[] = {1,2,3,4,5,6,7,8,9,10}
Output: 1 5
Explanation: The sum of elements
from 1st position to 5th position
is 15.
```

Your Task:
You don't need to read input or print anything. The task is to complete the function **subarraySum**() which takes arr, N and S as input parameters and returns a list containing the starting and ending positions of the first such occurring subarray from the left where sum equals to S. The two indexes in the list should be according to 1-based indexing. If no such subarray is found, return an array consisting only one element that is -1.

Move all negative elements to end

Easy Accuracy: 50.06% Submissions: 18954 Points: 2

Given an unsorted array **arr[]** of size **N** having both negative and positive integers. The task is place all negative element at the end of array without changing the order of positive element and negative element.

Example 1:

Input :
N = 8
arr[] = {1, -1, 3, 2, -7, -5, 11, 6 }
Output :
1 3 2 11 6 -1 -7 -5

Example 2:

Input :
N=8
arr[] = {-5, 7, -3, -4, 9, 10, -1, 11}
Output :
7 9 10 11 -5 -3 -4 -1

Your Task:
You don't need to read input or print anything. Your task is to complete the function **segregateElements()** which takes the array **arr[]** and its size **N** as inputs and **store** the answer in the array **arr[]** itself.

Union of two arrays

Basic Accuracy: 52.81% Submissions: 97197 Points: 1

Given two arrays **a[]** and **b[]** of size **n** and **m** respectively. The task is to find union between these two arrays.

Union of the two arrays can be defined as the set containing distinct elements from both the arrays. If there are repetitions, then only one occurrence of element should be printed in the union.

Example 1:

```
Input:
5 3
1 2 3 4 5
1 2 3
Output:
5
Explanation:
1, 2, 3, 4 and 5 are the
elements which comes in the union set
of both arrays. So count is 5.
```

Example 2:

```
Input:
6 2
85 25 1 32 54 6
85 2
Output:
7
Explanation:
85, 25, 1, 32, 54, 6, and
2 are the elements which comes in the
union set of both arrays. So count is 7.
```

Your Task:
Complete **doUnion** funciton that takes **a, n, b, m as parameters and returns** the count of union elements of the two arrays. The **printing** is done by the **driver** code.

Cyclically rotate an array by one

Basic Accuracy: 64.05% Submissions: 76745 Points: 1

Given an array, rotate the array by one position in clock-wise direction.

Example 1:

```
Input:
N = 5
A[] = {1, 2, 3, 4, 5}
Output:
5 1 2 3 4
```

Example 2:

```
Input:
N = 8
A[] = {9, 8, 7, 6, 4, 2, 1, 3}
Output:
3 9 8 7 6 4 2 1
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **rotate()** which takes the array **A[]** and its size **N** as inputs and modify the array.

Missing number in array

Easy Accuracy: 42.51% Submissions: 100k+ Points: 2

Given an array of size **N-1** such that it only contains distinct integers in the range of **1 to N**. Find the missing element.

Example 1:

```
Input:
N = 5
A[] = {1,2,3,5}
Output: 4
```

Example 2:

```
Input:
N = 10
A[] = {6,1,2,8,3,4,7,10,5}
Output: 9
```

Your Task :
You don't need to read input or print anything. Complete the function **MissingNumber()** that takes array and N as input parameters and returns the value of the missing number.

Count pairs with given sum

Easy Accuracy: 41.59% Submissions: 93333 Points: 2

Given an array of **N** integers, and an integer **K**, find the number of pairs of elements in the array whose sum is equal to **K**.

Example 1:

```
Input:
N = 4, K = 6
arr[] = {1, 5, 7, 1}
Output: 2
Explanation:
arr[0] + arr[1] = 1 + 5 = 6
and arr[1] + arr[3] = 5 + 1 = 6.
```

Example 2:

```
Input:
N = 4, X = 2
arr[] = {1, 1, 1, 1}
Output: 6
Explanation:
Each 1 will produce sum 2 with any 1.
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **getPairsCount()** which takes **arr[], n** and **k** as input parameters and returns the number of pairs that have sum K.

Find duplicates in an array

Easy Accuracy: 20.69% Submissions: 100k+ Points: 2

Given an array a[] of size N which contains elements from 0 to N-1, you need to find all the elements occurring more than once in the given array.

Example 1:

```
Input:
N = 4
a[] = {0,3,1,2}
Output: -1
Explanation: N=4 and all elements from 0
to (N-1 = 3) are present in the given
array. Therefore output is -1.
```

Example 2:

```
Input:
N = 5
a[] = {2,3,1,2,3}
Output: 2 3
Explanation: 2 and 3 occur more than once
in the given array.
```

Your Task:
Complete the function **duplicates()** which takes array a[] and n as input as parameters and returns a list of elements that occur more than once in the given array in sorted manner. If no such element is found, return list containing [-1].

Quick Sort

Medium Accuracy: 46.8% Submissions: 74282 Points: 4

Quick Sort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot.
Given an array arr[], its starting position low and its ending position high.

Implement the partition() and quickSort() functions to sort the array.

Example 1:

```
Input:
N = 5
arr[] = { 4, 1, 3, 9, 7}
Output:
1 3 4 7 9
```

Example 2:

```
Input:
N = 9
arr[] = { 2, 1, 6, 10, 4, 1, 3, 9, 7}
Output:
1 1 2 3 4 6 7 9 10
```

Your Task:
You don't need to read input or print anything. Your task is to complete the functions **partition()** and **quickSort()** which takes the array arr[], low and high as input parameters and partitions the array. Consider the last element as the pivot such that all the elements less than(or equal to) the pivot lie before it and the elements greater than it lie after the pivot.

Common elements

Easy Accuracy: 38.69% Submissions: 89661 Points: 2

Given three arrays sorted in increasing order. Find the elements that are common in all three arrays.
Note: can you take care of the duplicates without using any additional Data Structure?

Example 1:

```
Input:
n1 = 6; A = {1, 5, 10, 20, 40, 80}
n2 = 5; B = {6, 7, 20, 80, 100}
n3 = 8; C = {3, 4, 15, 20, 30, 70, 80, 120}
Output: 20 80
Explanation: 20 and 80 are the only
common elements in A, B and C.
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **commonElements()** which take the 3 arrays A[], B[], C[] and their respective sizes n1, n2 and n3 as inputs and returns an array containing the common element present in all the 3 arrays in sorted order.
If there are no such elements return an empty array. In this case the output will be printed as -1.

First Repeating Element

Easy Accuracy: 48.62% Submissions: 57450 Points: 2

Given an array arr[] of size n, find the first repeating element. The element should occurs more than once and the index of its first occurrence should be the smallest.

Example 1:

```
Input:
n = 7
arr[] = {1, 5, 3, 4, 3, 5, 6}
Output: 2
Explanation:
5 is appearing twice and
its first appearence is at index 2
which is less than 3 whose first
occuring index is 3.
```

Example 2:

```
Input:
n = 4
arr[] = {1, 2, 3, 4}
Output: -1
Explanation:
All elements appear only once so
answer is -1.
```

Your Task:
You don't need to read input or print anything. Complete the function **firstRepeated()** which takes arr and n as input parameters and return the position of the first repeating element. If there is no such element, return -1.
The position you return should be according to 1-based indexing.

Non-Repeating Element

Easy Accuracy: 51.48% Submissions: 10271 Points: 2

Find the first non-repeating element in a given array **arr** of **N** integers.
Note: Array consists of only positive and negative integers and **not zero**.

Example 1:

```
Input : arr[] = {-1, 2, -1, 3, 2}
Output : 3
Explanation:
-1 and 2 are repeating whereas 3 is
the only number occuring once.
Hence, the output is 3.
```

Example 2:

```
Input : arr[] = {1, 1, 1}
Output : 0
```

Your Task:
This is a function problem. The input is already taken care of by the driver code. You only need to complete the function **firstNonRepeating()** that takes an array **(arr)**, sizeOfArray **(n)**, and **returns** the first non-repeating element. The driver code takes care of the printing.

Subarrays with equal 1s and 0s

Medium Accuracy: 50.04% Submissions: 23512 Points: 4

Given an array containing 0s and 1s. Find the number of subarrays having equal number of 0s and 1s.

Example 1:

```
Input:
n = 7
A[] = {1,0,0,1,0,1,1}
Output: 8
Explanation: The index range for the 8
sub-arrays are: (0, 1), (2, 3), (0, 3), (3, 4),
(4, 5) ,(2, 5), (0, 5), (1, 6)
```

Example 2:

```
Input:
n = 5
A[] = {1,1,1,1,0}
Output: 1
Explanation: The index range for the
subarray is (3,4).
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **countSubarrWithEqualZeroAndOne()** which takes the array arr[] and the size of the array as inputs and returns the number of subarrays with equal number of 0s and 1s.

Alternate positive and negative numbers

Easy Accuracy: 49.41% Submissions: 32075 Points: 2

Given an unsorted array **Arr** of **N** positive and negative numbers. Your task is to create an array of alternate positive and negative numbers without changing the relative order of positive and negative numbers.
Note: Array should start with positive number.

Example 1:

```
Input:
N = 9
Arr[] = {9, 4, -2, -1, 5, 0, -5, -3, 2}
Output:
9 -2 4 -1 5 -5 0 -3 2
```

Example 2:

```
Input:
N = 10
Arr[] = {-5, -2, 5, 2, 4, 7, 1, 8, 0, -8}
Output:
5 -5 2 -2 4 -8 7 1 8 0
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **rearrange()** which takes the array of integers **arr[]** and **n** as parameters. You need to modify the array itself.

Subarray with 0 sum

Easy Accuracy: 49.91% Submissions: 81791 Points: 2

Given an array of positive and negative numbers. Find if there is a **subarray** (of size at-least one) with **0 sum**.

Example 1:

```
Input:
5
4 2 -3 1 6

Output:
Yes

Explanation:
2, -3, 1 is the subarray
with sum 0.
```

Example 2:

```
Input:
5
4 2 0 1 6

Output:
Yes

Explanation:
0 is one of the element
in the array so there exist a
subarray with sum 0.
```

Your Task:
You only need to complete the function **subArrayExists()** that takes **array and n** as **parameters** and **returns** true or false depending upon whether there is a subarray present with 0-sum or not. Printing will be taken care by the drivers code.

Factorials of large numbers

Medium Accuracy: 51.61% Submissions: 32044 Points: 4

Given an integer N, find its factorial.

Example 1:

```
Input: N = 5
Output: 120
Explanation : 5! = 1*2*3*4*5 = 120
```

Example 2:

```
Input: N = 10
Output: 3628800
Explanation :
10! = 1*2*3*4*5*6*7*8*9*10 = 3628800
```

Your Task:
You don't need to read input or print anything. Complete the function *factorial()* that takes integer N as input parameter and returns a list of integers denoting the digits that make up the factorial of N.

Stock span problem

Medium Accuracy: 49.89% Submissions: 53319 Points: 4

The stock span problem is a financial problem where we have a series of **n** daily price quotes for a stock and we need to calculate the span of stock's price for all **n** days.
The span S_i of the stock's price on a given day **i** is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than or equal to its price on the given day.
For example, if an array of 7 days prices is given as {100, 80, 60, 70, 60, 75, 85}, then the span values for corresponding 7 days are {1, 1, 1, 2, 1, 4, 6}.

Example 1:

Input:
N = 7, price[] = [100 80 60 70 60 75 85]
Output:
1 1 1 2 1 4 6
Explanation:
Traversing the given input span for 100 will be 1, 80 is smaller than 100 so the span is 1, 60 is smaller than 80 so the span is 1, 70 is greater than 60 so the span is 2 and so on. Hence the output will be 1 1 1 2 1 4 6.

Example 2:

Input:
N = 6, price[] = [10 4 5 90 120 80]
Output:
1 1 2 4 5 1
Explanation:
Traversing the given input span for 10 will be 1, 4 is smaller than 10 so the span will be 1, 5 is greater than 4 so the span will be 2 and so on. Hence, the output will be 1 1 2 4 5 1.

Video link for understanding
https://www.youtube.com/watch?v=-IFmgue8sF0

User Task:
The task is to complete the function **calculateSpan**() which takes two parameters, an array **price[]** denoting the price of stocks, and an integer **N** denoting the size of the array and number of days. This function finds the span of stock's price for all N days and returns an array of length **N** denoting the span for the i-th day.

Row with max 1s

Medium Accuracy: 42.51% Submissions: 73205 Points: 4

Given a boolean 2D array of n x m dimensions where each row is sorted. Find the 0-based index of the first row that has the maximum number of **1's**.

Example 1:

```
Input:
N = 4 , M = 4
Arr[][] = {{0, 1, 1, 1},
           {0, 0, 1, 1},
           {1, 1, 1, 1},
           {0, 0, 0, 0}}
Output: 2
Explanation: Row 2 contains 4 1's (0-based
indexing).
```

Example 2:

```
Input:
N = 2, M = 2
Arr[][] = {{0, 0}, {1, 1}}
Output: 1
Explanation: Row 1 contains 2 1's (0-based
indexing).
```

Your Task:

You don't need to read input or print anything. Your task is to complete the function **rowWithMax1s()** which takes the array of booleans **arr[][], n** and **m** as input parameters and returns the 0-based index of the first row that has the most number of 1s. If no such row exists, return -1.

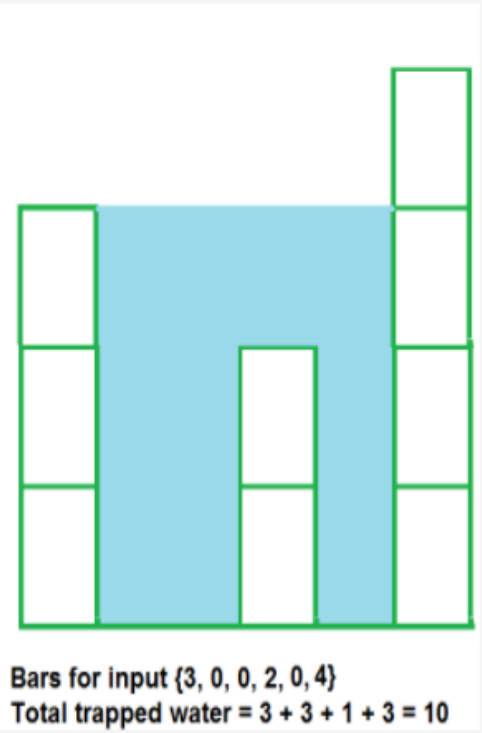
Trapping Rain Water

Medium Accuracy: 49.62% Submissions: 100k+ Points: 4

Given an array **arr[]** of **N** non-negative integers representing the height of blocks. If width of each block is 1, compute how much water can be trapped between the blocks during the rainy season.

Example 1:

Input:
N = 6
arr[] = {3,0,0,2,0,4}
Output:
10
Explanation:



Bars for input {3, 0, 0, 2, 0, 4}
Total trapped water = 3 + 3 + 1 + 3 = 10

Example 2:

Input:
N = 4
arr[] = {7,4,0,9}
Output:
10
Explanation:
Water trapped by above
block of height 4 is 3 units and above
block of height 0 is 7 units. So, the
total unit of water trapped is 10 units.

Example 3:

Input:
N = 3
arr[] = {6,9,9}
Output:
0
Explanation:
No water will be trapped.

Your Task:
You don't need to read input or print anything. The task is to complete the function **trappingWater**() which takes arr[] and N as input parameters and returns the total amount of water that can be trapped.