**1**

| *### 1st Problem Description* Given an integer array A of size N and an integer B, you have to return the same array after rotating it B times towards the right. Problem Constraints 1 <= N <= 105 1 <= A[i] <=109 1 <= B <= 109 Input Format The first argument given is the integer array A. The second argument given is the integer B. Output Format Return the array A after rotating it B times to the right Example Input Input 1: A = [1, 2, 3, 4] B = 2 Input 2: A = [2, 5, 6] B = 1 Example Output Output 1: [3, 4, 1, 2] Output 2: [6, 2, 5] Example Explanation Explanation 1: Rotate towards the right 2 times - [1, 2, 3, 4] => [4, 1, 2, 3] => [3, 4, 1, 2] Explanation 2: Rotate towards the right 1 time - [2, 5, 6] => [6, 2, 5] |
| --- |

| *# SOLUTION* *# 2nd problem solving ; if it should return the same array with the same order after many rotations* *# TC=O(N\*N)=O(N^2) and SC=O(N)* def rotating(\*data):  arr=list(data)  N=len(arr)  if 1 <=N<= 105:  for j in range(N): *# no. of rotations*  reference=arr[-1]  for i in range(N-1,0,-1):  if 1<=arr[i]<=109:  arr[i]=arr[i-1]  arr[0]=reference  *# print(arr)*  return arr   a=rotating(1,2,3,4,5,6,7,8) print(a)  *# 2nd option: to be rotated B times TC=O(N^2) and SC=O(N)* *# by thinking* def rotating(\*data,no):  arr=list(data)  N=len(arr)  if 1 <=N<= 105:  for j in range(no): *# no. of rotations*  reference=arr[-1]  for i in range(N-1,0,-1):  if 1<=arr[i]<=109:  arr[i]=arr[i-1]  arr[0]=reference  *# print(arr)*  return arr   a=rotating(1,2,3,4,5,6,7,8,no=2) print(a)  *# more optimised solution TC=O(1) and SC=O(N)* *# here it is found by researching* def rotating(\*data,no):  arr = list(data)  N = len(arr)  if 1 <= N <= 105:  num\_rotations = no % N *# number of rotations to perform*  print(num\_rotations)  arr = arr[-num\_rotations:] + arr[:-num\_rotations] *# perform rotations using slicing*  return arr  a=rotating(1,2,3,4,5,6,7,8,no=2) print(a) |
| --- |

**2**

| *### 2nd Problem Description:*  You are given a constant array A. You are required to return another array which is the reversed form of the input array. Problem Constraints ```1 <= A.size() <= 10000```  ```1 <= A[i] <= 10000``` Input Format First argument is a constant array A. Output Format Return an integer array. Example Input Input 1:  ```A = [1,2,3,2,1]```  Input 2:  ```A = [1,1,10]``` Example Output Output 1:  ```[1,2,3,2,1]``` Output 2: ```[10,1,1]``` |
| --- |

| *# SOLUTION* *# return the reverse of another array* def reverse(\*data): *#TC= O(N) and SC=O(N)*    arr=list(data)  arr\_2=[]  if 1<=len(arr)<=10000:  for i in range(len(arr)):  if 1<=arr[i]<=10000:  arr\_2.append(arr[i])  i=i+1  arr\_2.reverse()  return arr\_2  a=reverse(1,2,3,4) print(a) |
| --- |

**3**

| *### 3rd Problem Description* You are given an integer array A. You have to find the second largest element/value in the array or report that no such element exists. Problem Constraints 1 <= |A| <= 105 0 <= A[i] <= 109 Input Format The first argument is an integer array A. Output Format Return the second largest element. If no such element exist then return -1. Example Input Input 1:  A = [2, 1, 2] Input 2:  A = [2] Example Output Output 1:  1 Output 2:  -1 Example Explanation Explanation 1:  First largest element = 2  Second largest element = 1 Explanation 2:  There is no second largest element in the array. |
| --- |

| *# SOLUTION*  *# 3rd problem solving : TC= O(N) and SC=O(1)  def second\_max(\*data):  import statistics  arr=list(data)  N=len(arr)  maximum=max(arr)  second=0 # print(maximum)  if 1 <= N <= 105:  for i in range(N-1):  if type(arr[i]) is int and 0 <= arr[i] <= 109:  if N==1 or statistics.mean(arr)==maximum:return -1  elif arr[i]==maximum: continue  elif arr[i]>second: second=arr[i]  return second  import time start\_time=time.time() a=second\_max(100,70,100,109,109,109,55,88,105,66,109) print(a) end\_time = time.time() print(end\_time-start\_time) #0.002282381057739258     # another option for optimising; without using libraries def second\_max(\*data):  arr=list(data)  N=len(arr)  maximum=max(arr)  second=0  if 1<= N <= 105:  for i in range(N-1):  if type(arr[i]) is int and 0 <= arr[i] <= 109:  if N==1 or sum(arr)/N==maximum:return -1  elif arr[i]==maximum: continue  elif arr[i]>second: second=arr[i]  return second   import time start\_time = time.time() a=second\_max(100,70,100,109,109,109,55,88,105,66,109) print(a) end\_time = time.time() print(end\_time-start\_time) #0.001001596450805664   # using quicksort definition : TC = O(NlogN) and SC= O(N) def quicksort(arr): #TC= O(NlogN) & SC=O(N)  if len(arr) < 2:  return arr    #choose the pivot element  pivot = arr[0]   #Partition the array in two sub arrays  lesser =[i for i in arr[1:] if i<=pivot]  right =[i for i in arr[1:] if i > pivot]   return quicksort(lesser)+[pivot]+quicksort(right) def second\_max(\*data): #TC=O(NlogN) & SC=O(N)  arr=list(data)  N=len(arr)  max\_value=109  if 1<= N <= 105 and all(x <= max\_value for x in data):  if sum(arr)/N!=max(arr):  a=quicksort(arr)  return a[N-2]  else: return -1  else:  return f'the value in array is more than {max\_value}'   import time start\_time=time.time() b=second\_max(100,70,100,109,109,109,55,88,105,66,109) print(b) end\_time=time.time() print(end\_time-start\_time) # 0.0009984970092773438 c=second\_max(109,108,110,60)  # CONCLUSION  # the third option would be the most optimal situation according to the constraints. - no library is used to call - using quick sort algorithm which is quick to sort data and give the o/p specially with large data sets - time taken to execute for it is the least* |
| --- |