**DAA Assignment -1**

**(Implements the following problems using C++ / Python)**

**--21071A6766**

1 .Given a row wise sorted matrix of size **R\*C** where R and C are always **odd**, find the median of the matrix. **5Marks**

**Test Case 1:**

**Input**:

R = 3, C = 3

M = [[1, 3, 5],

  [2, 6, 9],

  [3, 6, 9]]

**Output:** 5

**Explanation**: Sorting matrix elements gives

us {1,2,3,3,5,6,6,9,9}. Hence, 5 is median.

**Test Case 2:**

**Input:**

R = 3, C = 1

M = [[1], [2], [3]]

**Output:** 2

**Explanation**: Sorting matrix elements gives

us {1,2,3}. Hence, 2 is median.

* **Constraints:**  
  1 <= R, C <= 400  
  1 <= matrix[i][j] <= 2000

Code(python):

row=int(input('Enter number of rows: '))

col=int(input('Enter number of columns: '))

m=[]

print('Enter matrix elements: ')

#taking space separated integers as input for each row separately

for i in range(row):

a=list(map(int,input().split()))

m.append(a)

import numpy

median=numpy.median(m)

print(f'Median of elements of matrix is: {median}')

#End of program

Output-1:



Output-2:



2. Given the arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays that represent the arrival and departure times of trains that stop. **5Marks**

**Test case 1**

***Input:****arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}, dep[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}****Output:****3****Explanation:****There are at-most three trains at a time (time between 9:40 to 12:00)*

**Test case 2**

***Input:****arr[] = {9:00, 9:40}, dep[] = {9:10, 12:00}****Output:****1****Explanation:****Only one platform is needed.*

Code(python):

arrive=list(input('Enter arrival timings of trains: ').split())

dep=list(input('Enter departure timings of trains: ').split())

arrival=[]

departure=[]

for i in arrive:

i=list(i)

i.remove(':')

j=''.join(i)

j=int(j)

arrival.append(j)

for i in dep:

i=list(i)

i.remove(':')

j=''.join(i)

j=int(j)

departure.append(j)

n=len(arrival)

arrival.sort()

departure.sort()

platforms=1

req\_platforms=1

i,j=1,0

while i<n and j<n:

if arrival[i]<=departure[j]:

platforms+=1

i+=1

else:

platforms-=1

j+=1

req\_platforms=max(req\_platforms,platforms)

print(f'Required number of platforms is: {req\_platforms}')

#end of program

Output-1:



Output-2:

