Full Name:	ARE JOSEPH VELANGINI PRAVALLIKA
Email:	pravallikajoseph20@gmail.com
Test Name:	**Mock Test**
Taken On:	12 Aug 2025 10:39:57 IST
Time Taken:	33 min 5 sec/ 40 min
Invited by:	Ankush
Invited on:	12 Aug 2025 10:39:11 IST
Skills Score:	
Tags Score:	

100%

195/195

scored in **Mock Test** in 33 min 5 sec on 12 Aug 2025 10:39:57 IST

Tags Score:

Algorithms	195/195
Constructive Algorithms	90/90
Core CS	195/195
Easy	105/105
Greedy Algorithms	90/90
Medium	90/90
Problem Solving	195/195
Search	105/105
Sorting	105/105
problem-solving	195/195

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review it in detail here -

	Question Description	Time Taken	Score	Status
Q1	**Find the Median** > **Coding**	15 min 39 sec	105/ 105	⚠
Q2	**Flipping the Matrix** > **Coding**	16 min 11 sec	90/ 90	⚠

QUESTION 1

⚠

Needs Review

Find the Median > Coding | Sorting | Search | Algorithms | Easy | problem-solving | Core CS | Problem Solving |

The median of a list of numbers is essentially its middle element after sorting. The same number of elements occur after it as before. Given a list of numbers with an odd number of elements, find the median?

Example

$arr = [5, 3, 1, 2, 4]$

The sorted array $arr' = [1, 2, 3, 4, 5]$. The middle element and the median is 3.

Function Description

Complete the *findMedian* function in the editor below.

findMedian has the following parameter(s):
  - *int arr[n]:* an unsorted array of integers

Returns
  - *int:* the median of the array

Input Format

The first line contains the integer n, the size of arr.
The second line contains n space-separated integers $arr[i]$

Constraints

  - $1 \le n \le 1000001$
  - n is odd
  - $-10000 \le arr[i] \le 10000$

Sample Input 0

```
7
0 1 2 4 6 5 3
```

Sample Output 0

```
3
```

Explanation 0

The sorted $arr = [0, 1, 2, 3, 4, 5, 6]$. It's middle element is at $arr[3] = 3$.

CANDIDATE ANSWER

Language used: **C**

```c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int compare(const void *a, const void *b){
5      return (*(int *)a - *(int *)b);
6  }
7
8  int findMedian(int arr[], int n){
9      qsort(arr, n, sizeof(int), compare);
10     return arr[n/2];
11 }
12
13 int main(){
14     int n;
15     scanf("%d", &n);
```

```
16
17      int arr[n];
18      for(int i = 0; i<n; i++){
19          scanf("%d", &arr[i]);
20      }
21      printf("%d\n", findMedian(arr, n));
22      return 0;
23  }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✓ Success	0	0.0084 sec	7.25 KB
Testcase 2	Easy	Hidden case	✓ Success	35	0.0091 sec	7.38 KB
Testcase 3	Easy	Hidden case	✓ Success	35	0.0095 sec	7.38 KB
Testcase 4	Easy	Hidden case	✓ Success	35	0.0316 sec	7.36 KB

No Comments

QUESTION 2

⚠
Needs Review

Score 90

Flipping the Matrix › Coding   Algorithms   Medium   Greedy Algorithms   Constructive Algorithms

problem-solving   Core CS   Problem Solving

QUESTION DESCRIPTION

Sean invented a game involving a $2n \times 2n$ matrix where each cell of the matrix contains an integer. He can reverse any of its rows or columns any number of times. The goal of the game is to maximize the sum of the elements in the $n \times n$ submatrix located in the upper-left quadrant of the matrix.

Given the initial configurations for q matrices, help Sean reverse the rows and columns of each matrix in the best possible way so that the sum of the elements in the matrix's upper-left quadrant is maximal.

Example
$matrix = [[1, 2], [3, 4]]$

```
1 2
3 4
```

It is 2×2 and we want to maximize the top left quadrant, a 1×1 matrix. Reverse row 1:

```
1 2
4 3
```

And now reverse column 0:

```
4 2
1 3
```

The maximal sum is 4.

Function Description

Complete the *flippingMatrix* function in the editor below.

flippingMatrix has the following parameters:
- *int matrix[2n][2n]:* a 2-dimensional array of integers

Returns
- *int:* the maximum sum possible.

Input Format

The first line contains an integer q, the number of queries.

The next q sets of lines are in the following format:
- The first line of each query contains an integer, n.
- Each of the next $2n$ lines contains $2n$ space-separated integers $matrix[i][j]$ in row i of the matrix.

Constraints

- $1 \le q \le 16$
- $1 \le n \le 128$
- $0 \le matrix[i][j] \le 4096$, where $0 \le i, j < 2n$.

Sample Input

```
STDIN            Function
-----            --------
1                q = 1
2                n = 2
112 42 83 119    matrix = [[112, 42, 83, 119], [56, 125, 56, 49], \
56 125 56 49               [15, 78, 101, 43], [62, 98, 114, 108]]
15 78 101 43
62 98 114 108
```

Sample Output

```
414
```

Explanation

Start out with the following $2n \times 2n$ matrix:

$$matrix = \begin{bmatrix} 112 & 42 & 83 & 119 \\ 56 & 125 & 56 & 49 \\ 15 & 78 & 101 & 43 \\ 62 & 98 & 114 & 108 \end{bmatrix}$$

Perform the following operations to maximize the sum of the $n \times n$ submatrix in the upper-left quadrant:

2. Reverse column 2 ($[83, 56, 101, 114] \rightarrow [114, 101, 56, 83]$), resulting in the matrix:

$$matrix = \begin{bmatrix} 112 & 42 & 114 & 119 \\ 56 & 125 & 101 & 49 \\ 15 & 78 & 56 & 43 \\ 62 & 98 & 83 & 108 \end{bmatrix}$$

3. Reverse row 0 ($[112, 42, 114, 119] \rightarrow [119, 114, 42, 112]$), resulting in the matrix:

$$matrix = \begin{bmatrix} 119 & 114 & 42 & 112 \\ 56 & 125 & 101 & 49 \\ 15 & 78 & 56 & 43 \\ 62 & 98 & 83 & 108 \end{bmatrix}$$

The sum of values in the $n \times n$ submatrix in the upper-left quadrant is $119 + 114 + 56 + 125 = 414$.

.

CANDIDATE ANSWER

Language used: **C**

```c
1  #include <stdio.h>
```

```c
int flippingMatrix(int n, int matrix[256][256]){
    int total = 0;
    int size = 2 * n;

    for (int i = 0; i<n; i++){
        for(int j = 0; j<n; j++){
            int a = matrix[i][j];
            int b = matrix[i][size -1 - j];
            int c = matrix[size - 1 - i][j];
            int d = matrix[size - 1 - i][size - 1 - j];

            int maxVal = a;
            if(b > maxVal)maxVal = b;
            if(c > maxVal)maxVal = c;
            if(d > maxVal)maxVal = d;

            total += maxVal;
        }
    }
    return total;
}

int main(){
    int q;
    scanf("%d", &q);

    while(q--){
        int n;
        scanf("%d", &n);

        int matrix[256][256];
        for(int i = 0; i<2 * n; i++){
            for(int j = 0; j<2 * n; j++){
                scanf("%d", &matrix[i][j]);
            }
        }
        int result = flippingMatrix(n, matrix);
        printf("%d\n", result);
    }
    return 0;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	⊘ Success	0	0.0089 sec	7.13 KB
Testcase 2	Easy	Hidden case	⊘ Success	15	0.0282 sec	7.25 KB
Testcase 3	Easy	Hidden case	⊘ Success	15	0.0411 sec	7.75 KB
Testcase 4	Easy	Hidden case	⊘ Success	15	0.0255 sec	7.5 KB
Testcase 5	Easy	Hidden case	⊘ Success	15	0.0478 sec	7.38 KB
Testcase 6	Easy	Hidden case	⊘ Success	15	0.0338 sec	7.63 KB
Testcase 7	Easy	Hidden case	⊘ Success	15	0.0638 sec	7.63 KB
Testcase 8	Easy	Sample case	⊘ Success	0	0.0067 sec	7.25 KB

No Comments