

<b>Started on</b>	Monday, 29 January 2024, 9:45 PM
<b>State</b>	Finished
<b>Completed on</b>	Monday, 29 January 2024, 10:25 PM
<b>Time taken</b>	40 mins 20 secs
<b>Marks</b>	30.00/30.00
<b>Grade</b>	10.00 out of 10.00 (100%)

**Question 1**

Correct

Mark 10.00 out of 10.00

An *array* is a type of data structure that stores elements of the same type in a contiguous block of memory. In an array,  $\mathbf{A}$ , of size  $\mathbf{N}$ , each memory location has some unique index,  $\mathbf{i}$  (where  $0 \leq \mathbf{i} < \mathbf{N}$ ), that can be referenced as  $\mathbf{A}[\mathbf{i}]$  or  $\mathbf{A}_i$ .

Reverse an array of integers.

**Note:** If you've already solved our C++ domain's *Arrays Introduction* challenge, you may want to skip this.

**Example**

$\mathbf{A} = [1, 2, 3]$

Return  $[3, 2, 1]$ .

**Function Description**

Complete the function *reverseArray* in the editor below.

*reverseArray* has the following parameter(s):

- *int A[n]*: the array to reverse

**Returns**

- *int[n]*: the reversed array

**Input Format**

The first line contains an integer,  $\mathbf{N}$ , the number of integers in  $\mathbf{A}$ .

The second line contains  $\mathbf{N}$  space-separated integers that make up  $\mathbf{A}$ .

**Constraints**

- $1 \leq \mathbf{N} \leq 10^3$
- $1 \leq \mathbf{A}[\mathbf{i}] \leq 10^4$ , where  $\mathbf{A}[\mathbf{i}]$  is the  $i^{th}$  integer in  $\mathbf{A}$

For example:

Input	Result
4 1 4 3 2	2 3 4 1
3 1 2 3	3 2 1

**Answer:** (penalty regime: 0 %)

Reset answer

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  string ltrim(const string &);
6  string rtrim(const string &);
7  vector<string> split(const string &);
8
9  /*
10 * Complete the 'reverseArray' function below.
11 *
12 * The function is expected to return an INTEGER_ARRAY.
13 * The function accepts INTEGER_ARRAY a as parameter.
14 */
15
16 vector<int> reverseArray(vector<int> a) {
17     int n = a.size();
18
19     for (int i=0; i<n/2;i++){//devsion gives the integer lower than float value
20         int tmp=a[i];//get the temporary value
21         a[i]=a[n-i-1];
22         a[n-i-1]=tmp;// interchange the array values
23     }
24
25 }
```

```
25     return a;
26
27 }
28
29 int main()
30 {
31     string arr_count_temp;
32     getline(cin, arr_count_temp);
33
34     int arr_count = stoi(ltrim(rtrim(arr_count_temp)));
35
36     string arr_temp_temp;
37     getline(cin, arr_temp_temp);
38
39     vector<string> arr_temp = split(rtrim(arr_temp_temp));
40
41     vector<int> arr(arr_count);
42
43     for (int i = 0; i < arr_count; i++) {
44         int arr_item = stoi(arr_temp[i]);
45
46         arr[i] = arr_item;
47     }
48
49     vector<int> res = reverseArray(arr);
50
51 }
```

	Input	Expected	Got	
✓	4 1 4 3 2	2 3 4 1	2 3 4 1	✓
✓	3 1 2 3	3 2 1	3 2 1	✓

Passed all tests! ✓

► [Show/hide question author's solution \(C++\).](#)

Correct

Marks for this submission: 10.00/10.00.

**Question 2**

Correct

Mark 10.00 out of 10.00

Given a  $6 \times 6$  2D Array, *arr*:

```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

An hourglass in *A* is a subset of values with indices falling in this pattern in *arr*'s graphical representation:

```
a b c
  d
e f g
```

There are **16** hourglasses in *arr*. An *hourglass sum* is the sum of an hourglass' values. Calculate the hourglass sum for every hourglass in *arr*, then print the *maximum* hourglass sum. The array will always be  $6 \times 6$ .

**Example**

*arr* =

```
-9 -9 -9 1 1 1
0 -9 0 4 3 2
-9 -9 -9 1 2 3
0 0 8 6 6 0
0 0 0 -2 0 0
0 0 1 2 4 0
```

The **16** hourglass sums are:

```
-63, -34, -9, 12,
-10, 0, 28, 23,
-27, -11, -2, 10,
9, 17, 25, 18
```

The highest hourglass sum is **28** from the hourglass beginning at row **1**, column **2**:

```
0 4 3
  1
8 6 6
```

**Note:** If you have already solved the Java domain's *Java 2D Array* challenge, you may wish to skip this challenge.

**Function Description**

Complete the function *hourglassSum* in the editor below.

*hourglassSum* has the following parameter(s):

- *int arr[6][6]*: an array of integers

**Returns**

- *int*: the maximum hourglass sum

**Input Format**

Each of the **6** lines of inputs *arr[i]* contains **6** space-separated integers *arr[i][j]*.

**Constraints**

- $-9 \leq arr[i][j] \leq 9$
- $0 \leq i, j \leq 5$

**Output Format**

Print the largest (maximum) hourglass sum found in *arr*.

**Sample Input**

```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 2 4 4 0
0 0 0 2 0 0
0 0 1 2 4 0
```

Sample Output

19

Explanation

*arr* contains the following hourglasses:

```
1 1 1 1 1 0 1 0 0 0 0 0
  1      0      0      0
1 1 1 1 1 0 1 0 0 0 0 0

0 1 0 1 0 0 0 0 0 0 0 0
  1      1      0      0
0 0 2 0 2 4 2 4 4 4 4 0

1 1 1 1 1 0 1 0 0 0 0 0
  0      2      4      4
0 0 0 0 0 2 0 2 0 2 0 0

0 0 2 0 2 4 2 4 4 4 4 0
  0      0      2      0
0 0 1 0 1 2 1 2 4 2 4 0
```

The hourglass with the maximum sum (**19**) is:

```
2 4 4
  2
1 2 4
```

For example:

Input	Result
1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 2 4 4 0 0 0 0 2 0 0 0 0 1 2 4 0	19

Answer: (penalty regime: 0 %)

Reset answer

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 string ltrim(const string &);
6 string rtrim(const string &);
7 vector<string> split(const string &);
8
9 /*
10  * Complete the 'hourglassSum' function below.
11  *
12  * The function is expected to return an INTEGER.
13  * The function accepts 2D_INTEGER_ARRAY arr as parameter.
14  */
15
16 int hourglassSum(vector<vector<int>> arr) {
17     int total = INT_MIN; // Initialize total to the lowest possible value
18     int sum=0 ;
19     for (int i = 0; i < 4; i++){//go through columns
20
21         for (int j = 0; j < 4; j++){ //go through rows
22             sum = arr[i][j]+arr[i][j+1]+arr[i][j+2]+arr[i+1][j+1]+arr[i+2][j]+arr[i+2][j+1]+arr[i+2][j+2];
23             //cout <<sum << endl;
24             if(sum >= total){// Update total if the current sum is greater
25                 total=sum;
26                 //cout << sum << endl;
27             }
28         }
29     }
30 }
```

```
29     }
30 }
31 return total;
32
33 }
34
35 int main()
36 {
37     vector<vector<int>> arr(6);
38
39     for (int i = 0; i < 6; i++) {
40         arr[i].resize(6);
41
42         string arr_row_temp_temp;
43         getline(cin, arr_row_temp_temp);
44
45         vector<string> arr_row_temp = split(rtrim(arr_row_temp_temp));
46
47         for (int j = 0; j < 6; j++) {
48             int arr_row_item = stoi(arr_row_temp[j]);
49
50             arr[i][j] = arr_row_item;
51         }
52     }
```

	Input	Expected	Got	
✓	1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 2 4 4 0 0 0 0 2 0 0 0 0 1 2 4 0	19	19	✓

Passed all tests! ✓

► [Show/hide question author's solution \(Cpp\).](#)

Correct

Marks for this submission: 10.00/10.00.

**Question 3**

Correct

Mark 10.00 out of 10.00

A *left rotation* operation on an array of size  $n$  shifts each of the array's elements **1** unit to the left. Given an integer,  $d$ , rotate the array that many steps left and return the result.

**Example** $d = 2$  $arr = [1, 2, 3, 4, 5]$ After **2** rotations,  $arr' = [3, 4, 5, 1, 2]$ .**Function Description**

Complete the `rotateLeft` function in the editor below.

`rotateLeft` has the following parameters:

- `int d`: the amount to rotate by
- `int arr[n]`: the array to rotate

**Returns**

- `int[n]`: the rotated array

**Input Format**

The first line contains two space-separated integers that denote  $n$ , the number of integers, and  $d$ , the number of left rotations to perform. The second line contains  $n$  space-separated integers that describe  $arr[]$ .

**Constraints**

- $1 \leq n \leq 10^5$
- $1 \leq d \leq n$
- $1 \leq a[i] \leq 10^6$

**Sample Input**

```
5 4
1 2 3 4 5
```

**Sample Output**

```
5 1 2 3 4
```

**Explanation**

To perform  $d = 4$  left rotations, the array undergoes the following sequence of changes:

$[1, 2, 3, 4, 5] \rightarrow [2, 3, 4, 5, 1] \rightarrow [3, 4, 5, 1, 2] \rightarrow [4, 5, 1, 2, 3] \rightarrow [5, 1, 2, 3, 4]$

For example:

Input	Result
5 4 1 2 3 4 5	5 1 2 3 4

**Answer:** (penalty regime: 0 %)

Reset answer

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 string ltrim(const string &);
6 string rtrim(const string &);
7 vector<string> split(const string &);
8
9 /*
10  * Complete the 'rotateLeft' function below.
11  *
12  * The function is expected to return an INTEGER_ARRAY.
13  * The function accepts following parameters:
```

```
14 * 1. INTEGER d
15 * 2. INTEGER_ARRAY arr
16 */
17
18 vector<int> rotateLeft(int d, vector<int> arr) {
19     int n = arr.size();
20     //int r =4;
21     //int tmp;
22     //int arr [n] ={1,2,3,4,5};
23     for (int i = 0; i < d; i++){
24         int tmp = arr[0]; //take first element as temporary value
25         for (int k =0; k <n-1; k++){
26             arr[k]=arr[k+1]; // assign the i +1 element to ith element
27         }
28         arr[n-1]=tmp;
29     }
30
31
32
33     return arr;
34 }
35
36
37 int main()
38 {
39     string first_multiple_input_temp;
40     getline(cin, first_multiple_input_temp);
41
42     vector<string> first_multiple_input = split(rtrim(first_multiple_input_temp));
43
44     int n = stoi(first_multiple_input[0]);
45
46     int d = stoi(first_multiple_input[1]);
47
48     string arr_temp_temp;
49     getline(cin, arr_temp_temp);
50
51     vector<string> arr_temp = split(rtrim(arr_temp_temp));
52 }
```

	Input	Expected	Got	
✓	5 4 1 2 3 4 5	5 1 2 3 4	5 1 2 3 4	✓

Passed all tests! ✓

► Show/hide question author's solution (Cpp).

Correct

Marks for this submission: 10.00/10.00.