

Started on	Monday, 29 January 2024, 9:45 PM
State	Finished
Completed on	Monday, 29 January 2024, 10:25 PM
Time taken	40 mins 29 secs
Marks	30.00/30.00
Grade	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, A , of size N , each memory location has some unique index, i (where $0 \leq i < N$), that can be referenced as $A[i]$ or 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

$A = [1, 2, 3]$

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, N , the number of integers in A .

The second line contains N space-separated integers that make up A .

Constraints

- $1 \leq N \leq 10^3$
- $1 \leq A[i] \leq 10^4$, where $A[i]$ is the i^{th} integer in 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();//getting the size of the array
18     for (int i=0;i<n/2;i++){//swapping numbers
19         int temp=a[i];
20         a[i]=a[n-1-i];
```

```

21         a[n-1-i]=temp;
22     }
23     return a;
24 }
25
26 int main()
27 {
28     string arr_count_temp;
29     getline(cin, arr_count_temp);
30
31     int arr_count = stoi(ltrim(rtrim(arr_count_temp)));
32
33     string arr_temp_temp;
34     getline(cin, arr_temp_temp);
35
36     vector<string> arr_temp = split(rtrim(arr_temp_temp));
37
38     vector<int> arr(arr_count);
39
40
41     for (int i = 0; i < arr_count; i++) {
42         int arr_item = stoi(arr_temp[i]);
43
44         arr[i] = arr_item;
45     }
46
47     vector<int> res = reverseArray(arr);
48
49     for (size_t i = 0; i < res.size(); i++) {
50         cout << res[i];
51
52         if (i != res.size() - 1) {

```

	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 6×6 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 6×6 .

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 max_sum=INT_MIN;
18     //calculate hourglass sums
19     for (int i=0;i<=3;i++){
20         for (int j=0;j<=3;j++){
21             int 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]
22             //storing the maximum value while comparing
23             if(max<sum){
24                 max=sum;
25             }
26         }
27     }
28     return max;
29 }
```

```

24         ++max_sum=sum;
25         max_sum=sum;
26     }
27 }
28
29 }
30
31 return max_sum;//returning the max_sum
32
33
34 }
35
36 int main()
37 {
38     vector<vector<int>> arr(6);
39
40     for (int i = 0; i < 6; i++) {
41         arr[i].resize(6);
42
43         string arr_row_temp_temp;
44         getline(cin, arr_row_temp_temp);
45
46         vector<string> arr_row_temp = split(rtrim(arr_row_temp_temp));
47
48         for (int j = 0; j < 6; j++) {
49             int arr_row_item = stoi(arr_row_temp[j]);
50
51             arr[i][j] = arr_row_item;
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
```

```

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();//assigning the size of the array
20     for (int k=d;k>0;k--){//doing d no.of rotations
21         int temp=arr[0];//storing the first value
22         //shift each element front by one place except the first element
23         for (int j=1;j<n;j++){
24             arr[j-1]=arr[j];
25         }
26         arr[n-1]=temp;//giving the correct location to temp
27     }
28     return arr;
29 }
30
31
32 int main()
33 {
34     string first_multiple_input_temp;
35     getline(cin, first_multiple_input_temp);
36
37     vector<string> first_multiple_input = split(rtrim(first_multiple_input_temp));
38
39     int n = stoi(first_multiple_input[0]);
40
41     int d = stoi(first_multiple_input[1]);
42
43     string arr_temp_temp;
44     getline(cin, arr_temp_temp);
45
46     vector<string> arr_temp = split(rtrim(arr_temp_temp));
47
48     vector<int> arr(n);
49
50     for (int i = 0; i < n; i++) {
51         int arr_item = stoi(arr_temp[i]);
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.