PROBLEM STATEMENT

HackerRank Practice > Data Structures > Arrays > 2D Array - DS

Problem | Submissions | Leaderboard | Discussions | Editorial

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.

Problem | Submissions | Leaderboard | Discussions

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
```

Leade | Discussions | Editorial

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
```

PROGRAM USED TO SOLVE THE PROBLEM STATEMENT

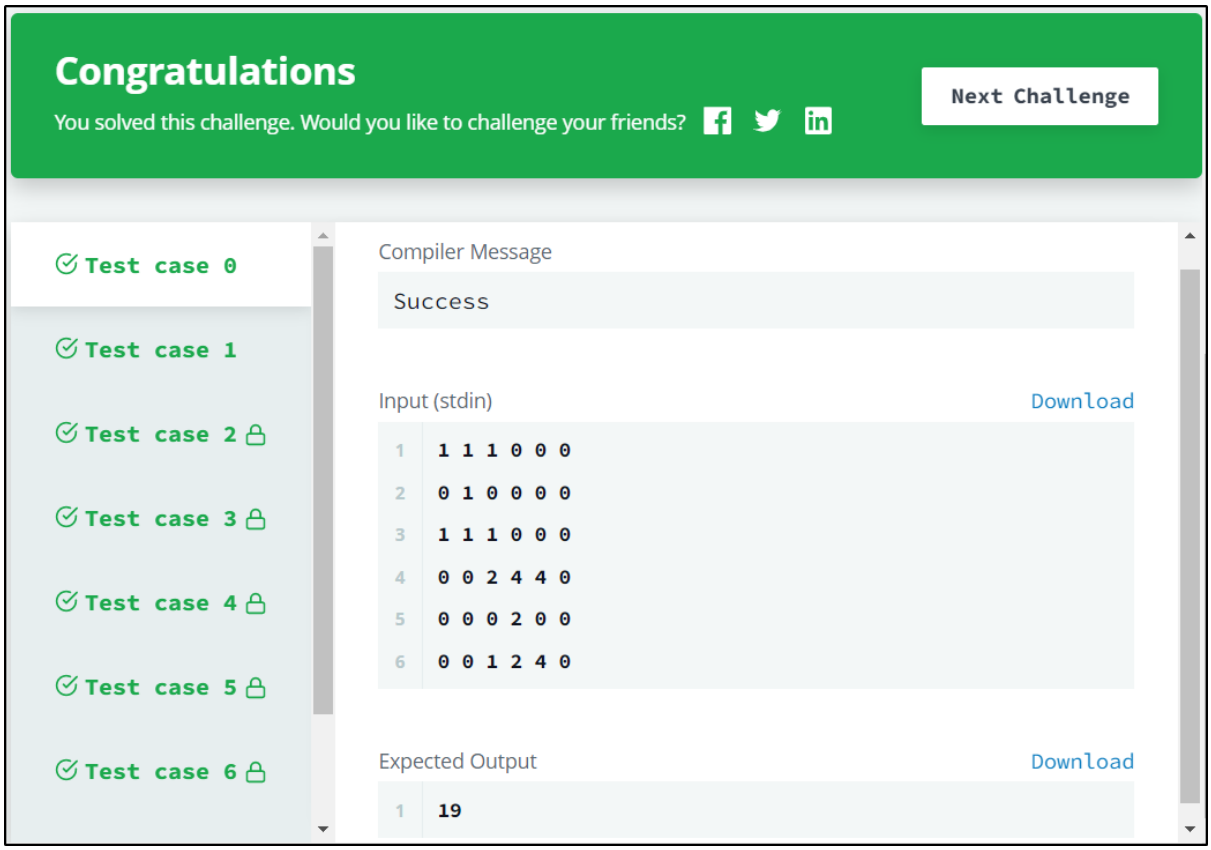
```
#include <assert.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char** split_string(char*);

// Complete the hourglassSum function below.
int hourglassSum(int arr_rows, int arr_columns, int** arr)
 {

   int sum = 0,max = -63;

   for(int i=0; i<4; i++){

   for(int j=0; j<4; j++){
       sum = 0;
       sum += arr[i][j] + arr[i][j+1] + arr[i][j+2];
// x y z
       sum += arr[i+1][j+1];
//   o
       sum += arr[i+2][j] + arr[i+2][j+1] + arr[i+2][j+2];
// a b c


           if(sum>max)
           max = sum;

          }


      }

 return max;
}
```

```c
int main()
{
    FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

    int** arr = malloc(6 * sizeof(int*));

    for (int i = 0; i < 6; i++) {
        *(arr + i) = malloc(6 * (sizeof(int)));

        char** arr_item_temp = split_string(readline());

        for (int j = 0; j < 6; j++) {
            char* arr_item_endptr;
            char* arr_item_str = *(arr_item_temp + j);
            int arr_item = strtol(arr_item_str, &arr_item_
endptr, 10);

            if (arr_item_endptr == arr_item_str || *arr_it
em_endptr != '\0') { exit(EXIT_FAILURE); }

            *(*(arr + i) + j) = arr_item;
        }
    }

    int arr_rows = 6;
    int arr_columns = 6;

    int result = hourglassSum(arr_rows, arr_columns, arr);

    fprintf(fptr, "%d\n", result);

    fclose(fptr);

    return 0;
}

char* readline() {
    size_t alloc_length = 1024;
    size_t data_length = 0;
    char* data = malloc(alloc_length);
```

```
    while (true) {
        char* cursor = data + data_length;
        char* line = fgets(cursor, alloc_length - data_len
gth, stdin);

        if (!line) { break; }

        data_length += strlen(cursor);

        if (data_length < alloc_length - 1 || data[data_le
ngth - 1] == '\n') { break; }

        size_t new_length = alloc_length << 1;
        data = realloc(data, new_length);

        if (!data) { break; }

        alloc_length = new_length;
    }

    if (data[data_length - 1] == '\n') {
        data[data_length - 1] = '\0';
    }

    data = realloc(data, data_length);

    return data;
}

char** split_string(char* str) {
    char** splits = NULL;
    char* token = strtok(str, " ");

    int spaces = 0;

    while (token) {
        splits = realloc(splits, sizeof(char*) * ++spaces)
;
        if (!splits) {
            return splits;
        }
```

```
        splits[spaces - 1] = token;

        token = strtok(NULL, " ");
    }

    return splits;
}
```

TEST CASES


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
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
Expected Output
Download
19