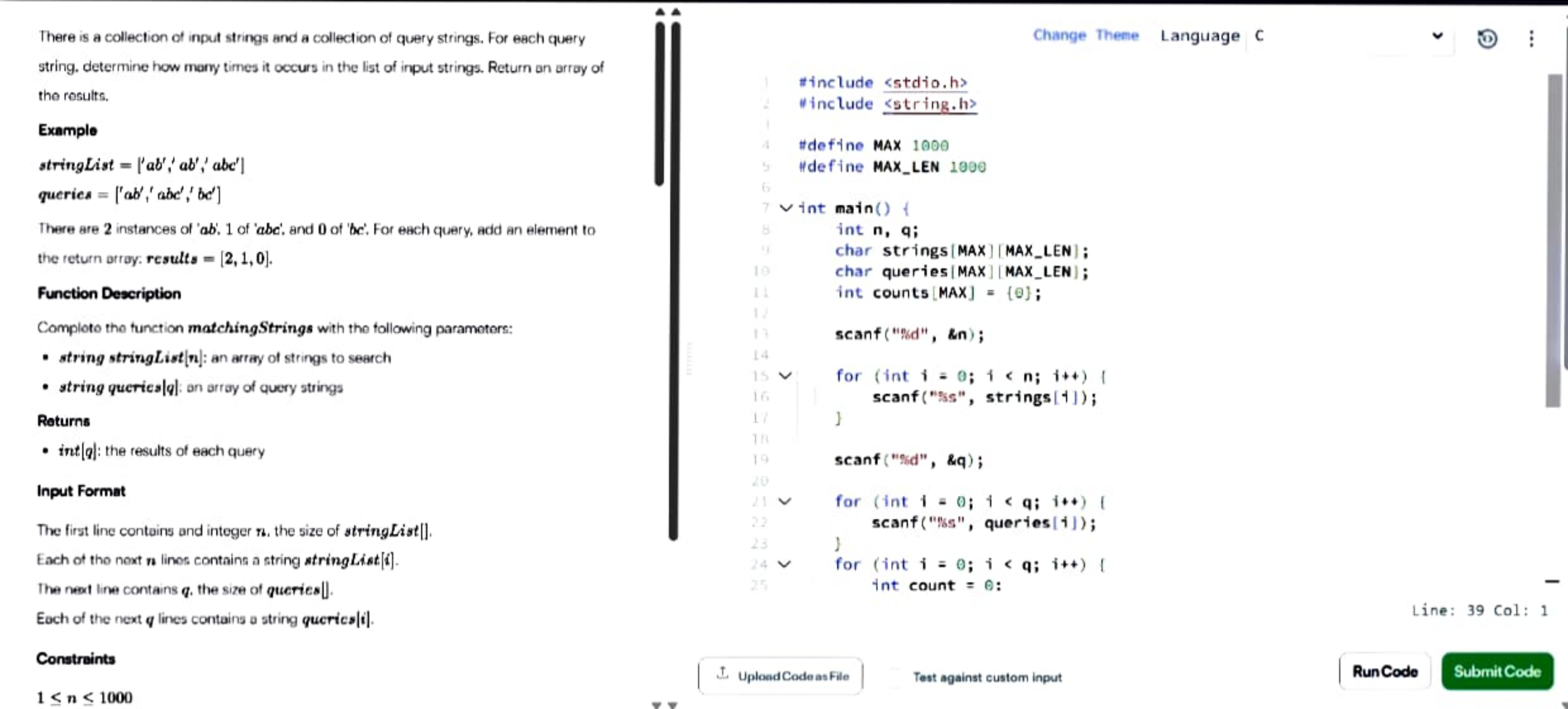


There is a collection of input strings and a collection of query strings. For each query string, determine how many times it occurs in the list of input strings. Return an array of the results.

Example

$stringList = ['ab', 'ab', 'abc']$

$queries = ['ab', 'abc', 'bc']$

There are 2 instances of '*ab*', 1 of '*abc*', and 0 of '*bc*'. For each query, add an element to the return array: $results = [2, 1, 0]$.

Function Description

Complete the function *matchingStrings* with the following parameters:

- *string stringList[n]*: an array of strings to search
- *string queries[q]*: an array of query strings

Returns

- *int[q]*: the results of each query

Input Format

The first line contains and integer n, the size of $stringList[]$.

Each of the next n lines contains a string $stringList[i]$.

The next line contains q, the size of $queries[]$.

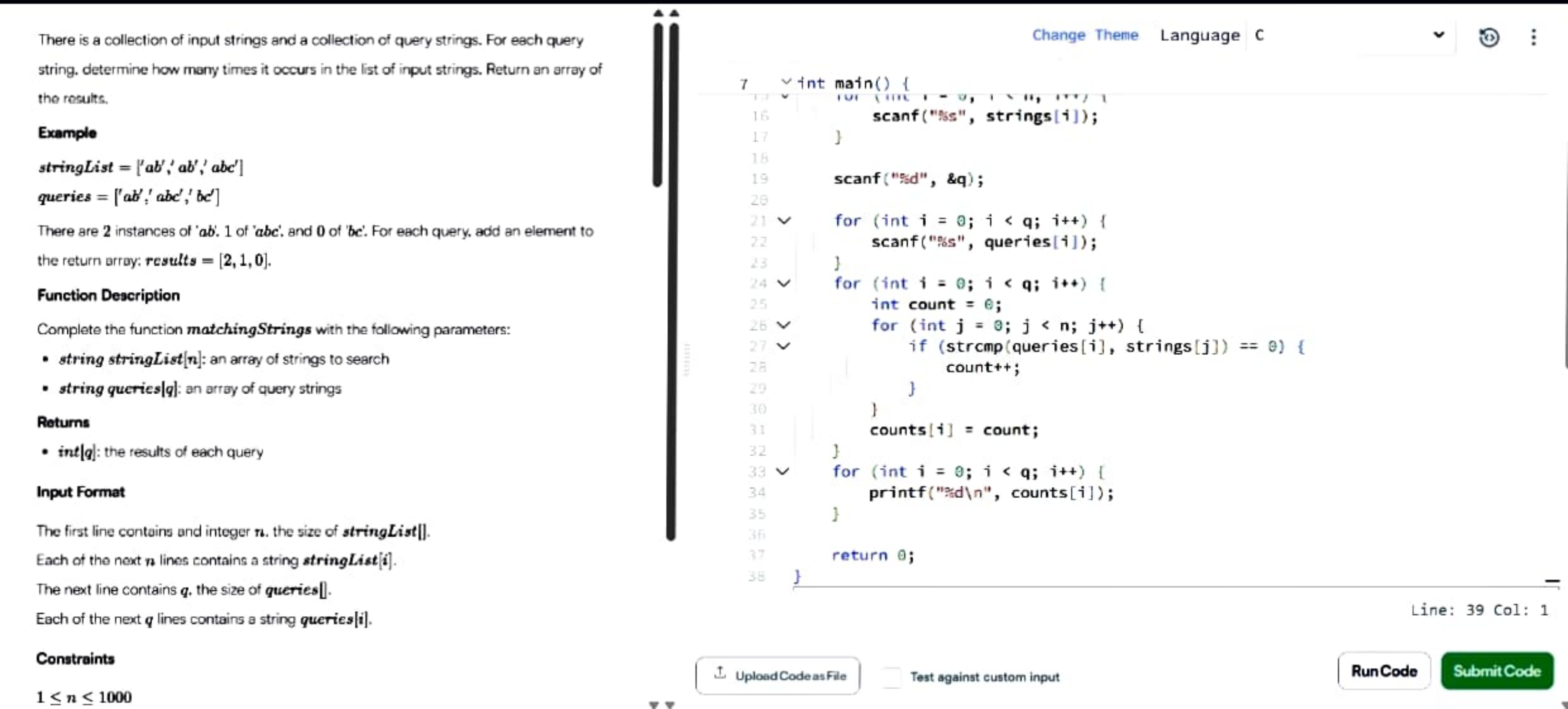
Each of the next q lines contains a string $queries[i]$.

Constraints

$1 \leq n \leq 1000$

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

#define MAX 1000
#define MAX_LEN 1000

int main() {
    int n, q;
    char strings[MAX][MAX_LEN];
    char queries[MAX][MAX_LEN];
    int counts[MAX] = {0};

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%s", strings[i]);
    }

    scanf("%d", &q);

    for (int i = 0; i < q; i++) {
        scanf("%s", queries[i]);
    }
    for (int i = 0; i < q; i++) {
        int count = 0;
```

Line: 39 Col: 1



There is a collection of input strings and a collection of query strings. For each query string, determine how many times it occurs in the list of input strings. Return an array of the results.

Example

$stringList = ['ab', 'ab', 'abc']$

$queries = ['ab', 'abc', 'bc']$

There are 2 instances of 'ab', 1 of 'abc', and 0 of 'bc'. For each query, add an element to the return array: $results = [2, 1, 0]$.

Function Description

Complete the function *matchingStrings* with the following parameters:

- *string stringList*$[n]$: an array of strings to search
- *string queries*$[q]$: an array of query strings

Returns

- *int*$[q]$: the results of each query

Input Format

The first line contains and integer n, the size of *stringList*$[]$.

Each of the next n lines contains a string *stringList*$[i]$.

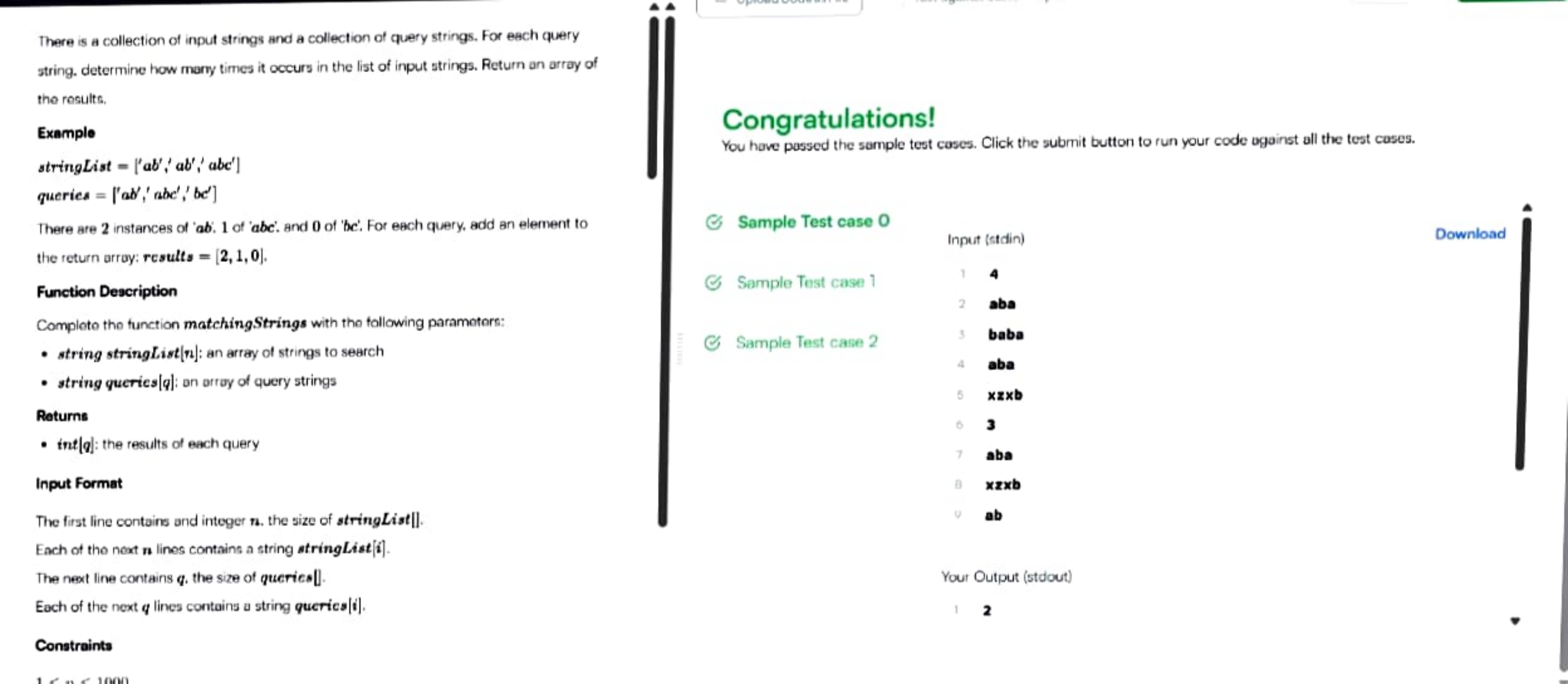
The next line contains q, the size of *queries*$[]$.

Each of the next q lines contains a string *queries*$[i]$.

Constraints

$1 \leq n \leq 1000$

```c
int main() {
    for (int i = 0; i < n; i++) {
        scanf("%s", strings[i]);
    }

    scanf("%d", &q);

    for (int i = 0; i < q; i++) {
        scanf("%s", queries[i]);
    }
    for (int i = 0; i < q; i++) {
        int count = 0;
        for (int j = 0; j < n; j++) {
            if (strcmp(queries[i], strings[j]) == 0) {
                count++;
            }
        }
        counts[i] = count;
    }
    for (int i = 0; i < q; i++) {
        printf("%d\n", counts[i]);
    }

    return 0;
}
```



There is a collection of input strings and a collection of query strings. For each query string, determine how many times it occurs in the list of input strings. Return an array of the results.

Example

$stringList = ['ab', 'ab', 'abc']$

$queries = ['ab', 'abc', 'bc']$

There are 2 instances of 'ab', 1 of 'abc', and 0 of 'bc'. For each query, add an element to the return array: $results = [2, 1, 0]$.

Function Description

Complete the function $matchingStrings$ with the following parameters:

- $string\ stringList[n]$: an array of strings to search
- $string\ queries[q]$: an array of query strings

Returns

- $int[q]$: the results of each query

Input Format

The first line contains and integer n, the size of $stringList[]$.

Each of the next n lines contains a string $stringList[i]$.

The next line contains q, the size of $queries[]$.

Each of the next q lines contains a string $queries[i]$.

Constraints

$1 \le n \le 1000$

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

✓ Sample Test case 1

✓ Sample Test case 2

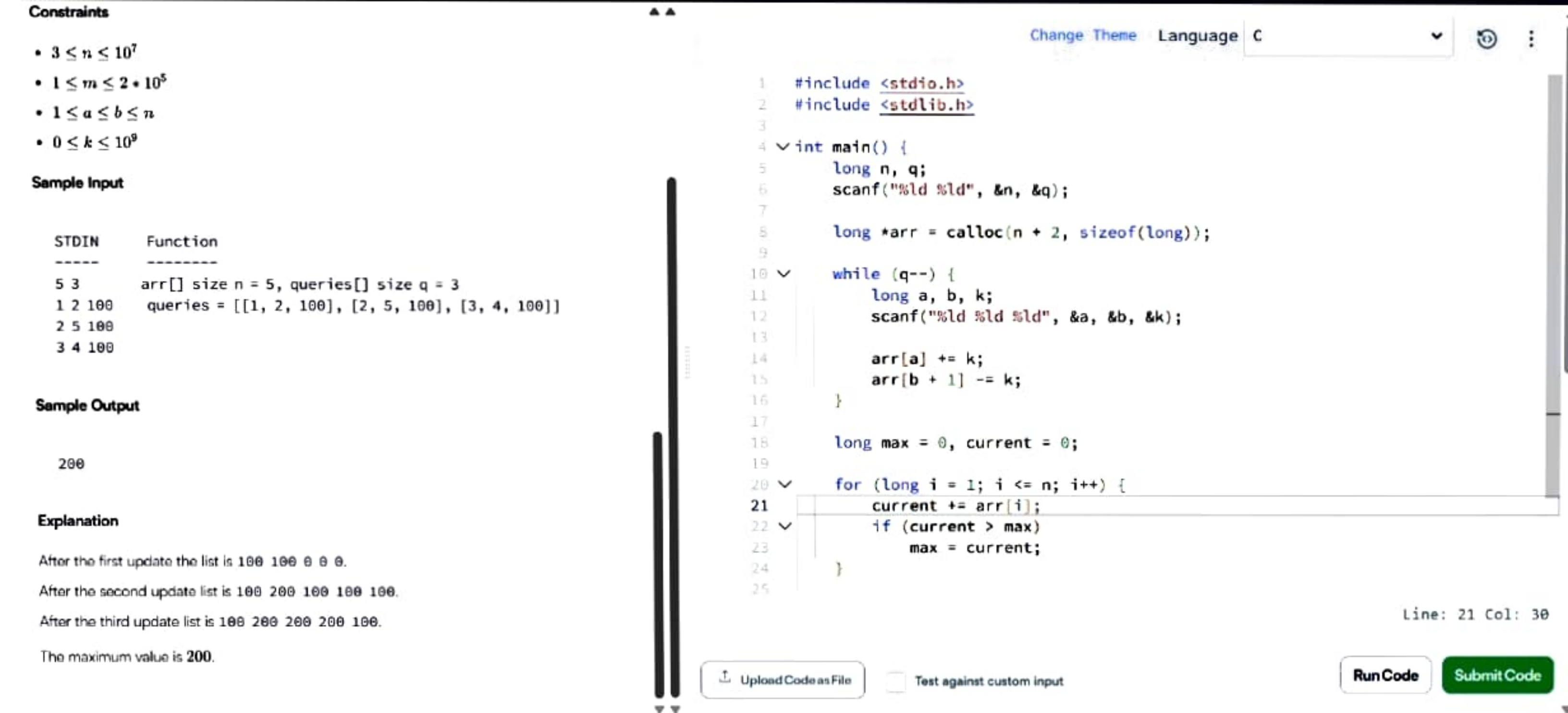
Input (stdin)

Download

```
4
aba
baba
aba
xzxb
3
aba
xzxb
ab
```

Your Output (stdout)

```
2
```



Constraints

- $3 \le n \le 10^7$
- $1 \le m \le 2 \cdot 10^5$
- $1 \le a \le b \le n$
- $0 \le k \le 10^9$

Sample Input

```
STDIN       Function
-----       --------
5 3        arr[] size n = 5, queries[] size q = 3
1 2 100     queries = [[1, 2, 100], [2, 5, 100], [3, 4, 100]]
2 5 100
3 4 100
```

Sample Output

```
200
```

Explanation

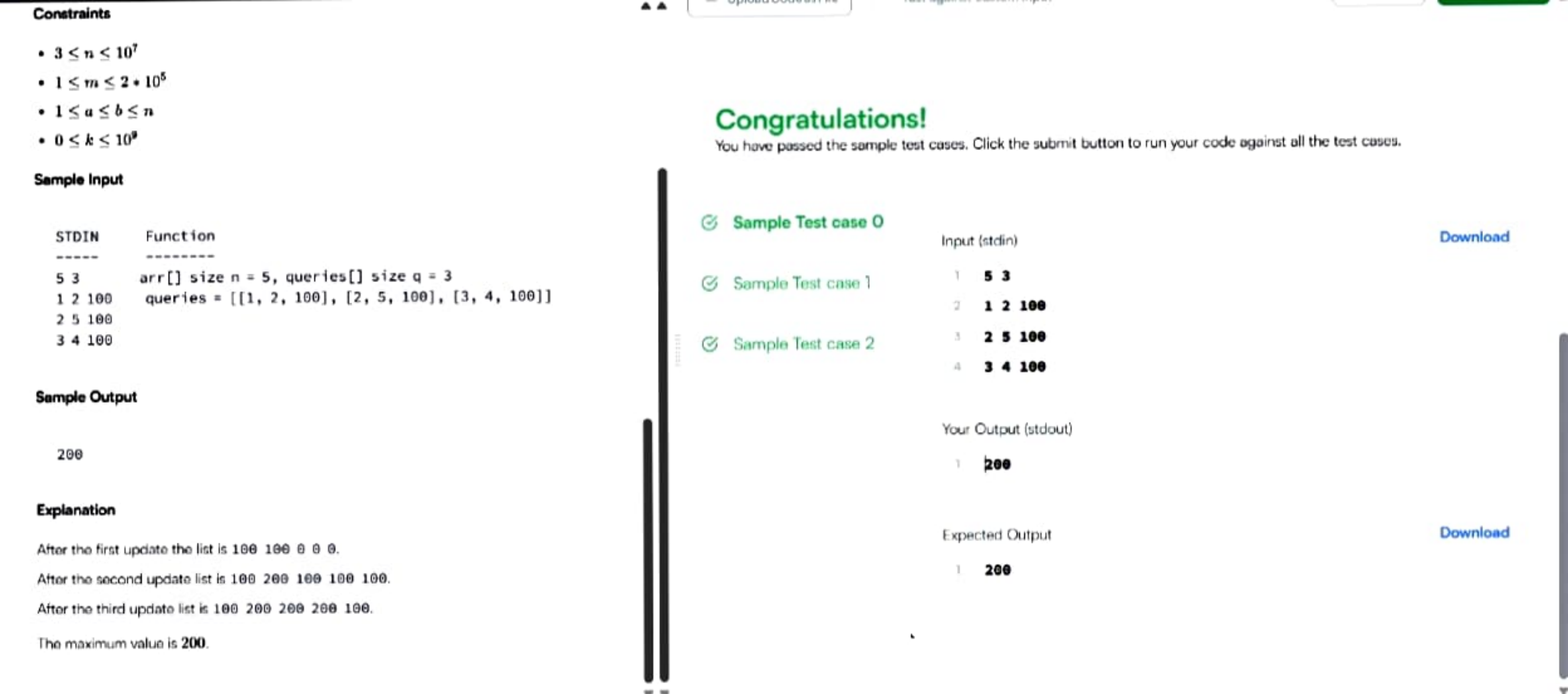
After the first update the list is `100 100 0 0 0`.

After the second update list is `100 200 100 100 100`.

After the third update list is `100 200 200 200 100`.

The maximum value is 200.

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

int main() {
    long n, q;
    scanf("%ld %ld", &n, &q);

    long *arr = calloc(n + 2, sizeof(long));

    while (q--) {
        long a, b, k;
        scanf("%ld %ld %ld", &a, &b, &k);

        arr[a] += k;
        arr[b + 1] -= k;
    }

    long max = 0, current = 0;

    for (long i = 1; i <= n; i++) {
        current += arr[i];
        if (current > max)
            max = current;
    }

```



Constraints

- $3 \leq n \leq 10^7$
- $1 \leq m \leq 2 * 10^5$
- $1 \leq a \leq b \leq n$
- $0 \leq k \leq 10^9$

Sample Input

```
STDIN       Function
-----       --------
5 3        arr[] size n = 5, queries[] size q = 3
1 2 100     queries = [[1, 2, 100], [2, 5, 100], [3, 4, 100]]
2 5 100
3 4 100
```

Sample Output

```
200
```

Explanation

After the first update the list is `100 100 0 0 0`.

After the second update list is `100 200 100 100 100`.

After the third update list is `100 200 200 200 100`.

The maximum value is **200**.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Input (stdin)

```
5 3
1 2 100
2 5 100
3 4 100
```

Your Output (stdout)

```
200
```

Expected Output

```
200
```

Sample Output

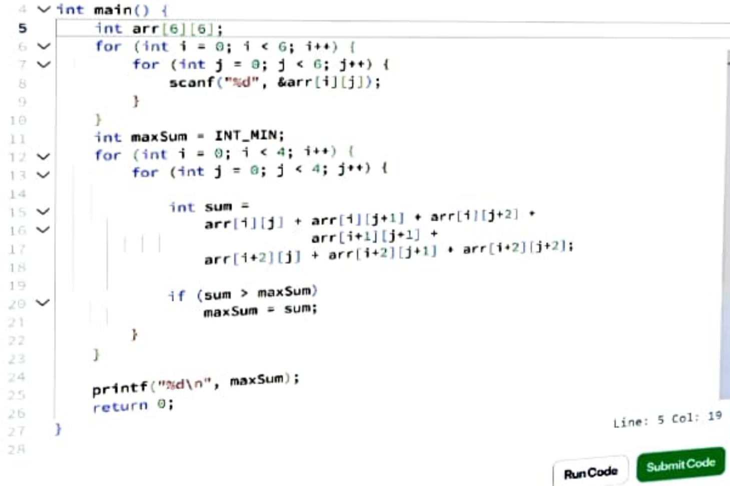
```
19
```

Explanation

arr contains the following hourglasses:

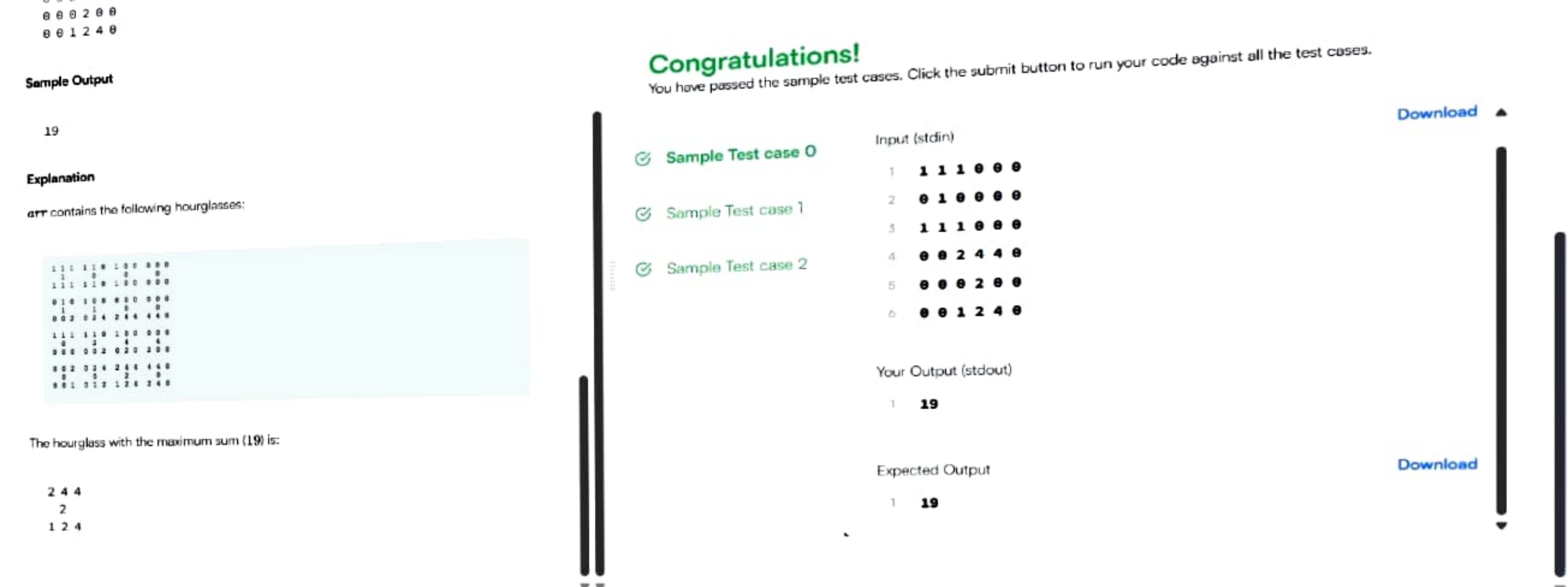
```
1 1 1   1 1 0   1 0 0   0 0 0
  1       0       0       0
1 1 1   1 1 0   1 0 0   0 0 0

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:



```c
int main() {
    int arr[6][6];
    for (int i = 0; i < 6; i++) {
        for (int j = 0; j < 6; j++) {
            scanf("%d", &arr[i][j]);
        }
    }
    int maxSum = INT_MIN;
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {

            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];

            if (sum > maxSum)
                maxSum = sum;
        }
    }

    printf("%d\n", maxSum);
    return 0;
}
```

Line: 5 Col: 19



```
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
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Sample Test case 2

Input (stdin) Download

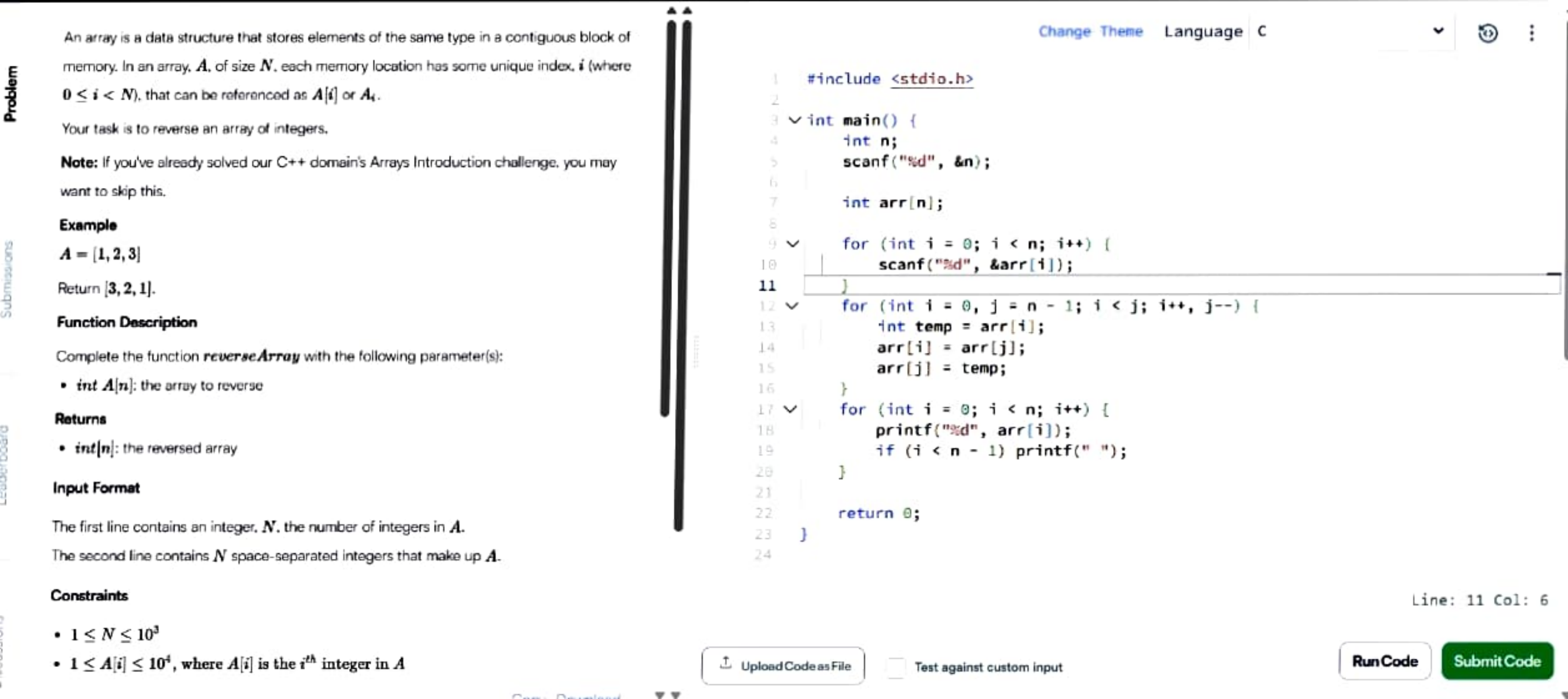
```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 2 4 4 0
0 0 0 2 0 0
0 0 1 2 4 0
```

Your Output (stdout)

```
19
```

Expected Output Download

```
19
```



An array is a 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.

Your task is to 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* with 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.

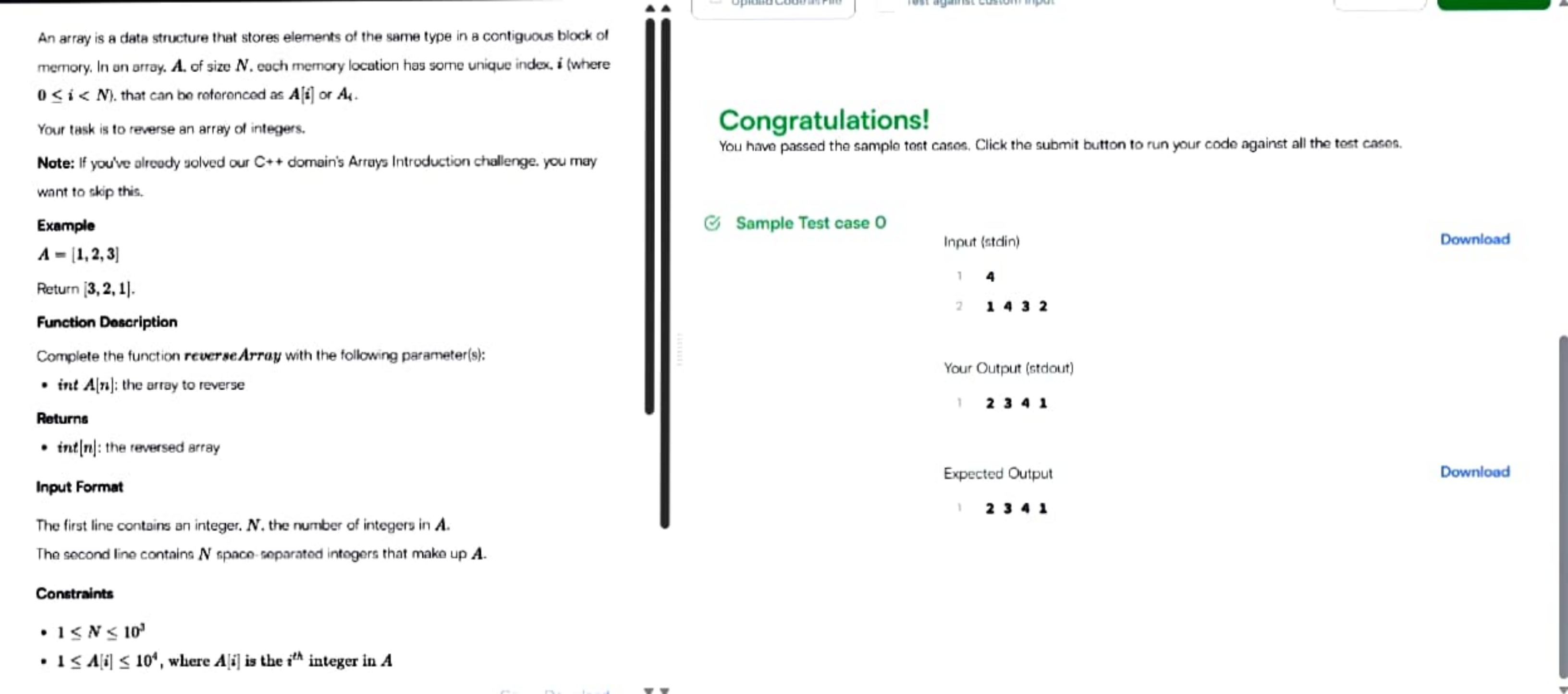
The second line contains N space-separated integers that make up A.

Constraints

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

```c
#include <stdio.h>

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

    int arr[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    for (int i = 0, j = n - 1; i < j; i++, j--) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
    for (int i = 0; i < n; i++) {
        printf("%d", arr[i]);
        if (i < n - 1) printf(" ");
    }

    return 0;
}
```

Line: 11 Col: 6



An array is a 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.

Your task is to 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 $reverseArray$ with 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

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

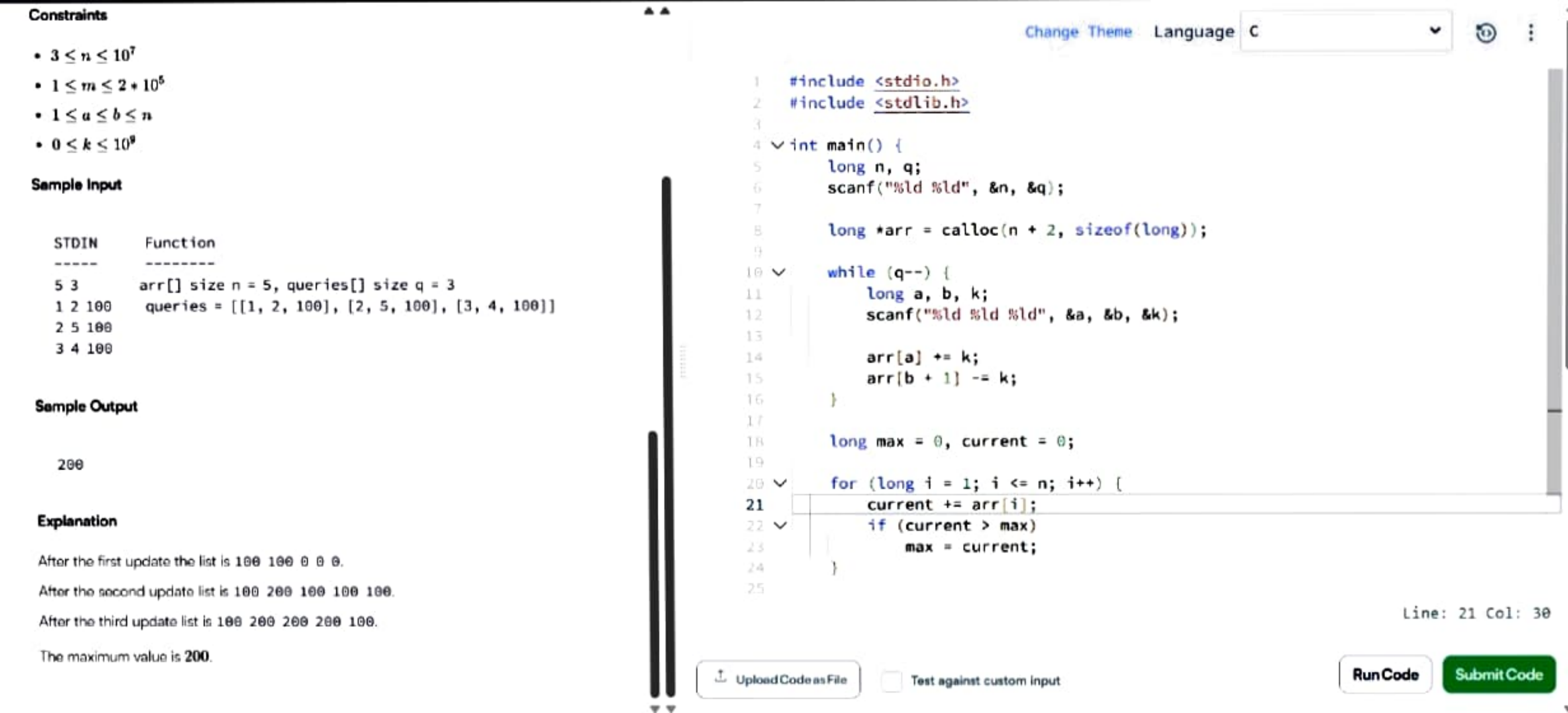
```
4
1 4 3 2
```

Your Output (stdout)

```
2 3 4 1
```

Expected Output Download

```
2 3 4 1
```



Constraints

- $3 \le n \le 10^7$
- $1 \le m \le 2 * 10^5$
- $1 \le a \le b \le n$
- $0 \le k \le 10^9$

Sample Input

```
STDIN       Function
-----       --------
5 3        arr[] size n = 5, queries[] size q = 3
1 2 100     queries = [[1, 2, 100], [2, 5, 100], [3, 4, 100]]
2 5 100
3 4 100
```

Sample Output

```
200
```

Explanation

After the first update the list is `100 100 0 0 0`.

After the second update list is `100 200 100 100 100`.

After the third update list is `100 200 200 200 100`.

The maximum value is **200**.

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

int main() {
    long n, q;
    scanf("%ld %ld", &n, &q);

    long *arr = calloc(n + 2, sizeof(long));

    while (q--) {
        long a, b, k;
        scanf("%ld %ld %ld", &a, &b, &k);

        arr[a] += k;
        arr[b + 1] -= k;
    }

    long max = 0, current = 0;

    for (long i = 1; i <= n; i++) {
        current += arr[i];
        if (current > max)
            max = current;
    }

```

Line: 21 Col: 30

Upload Code as File

Test against custom input

Constraints

- $3 \le n \le 10^7$
- $1 \le m \le 2 * 10^5$
- $1 \le a \le b \le n$
- $0 \le k \le 10^9$

Sample Input

```
STDIN       Function
-----       --------
5 3         arr[] size n = 5, queries[] size q = 3
1 2 100      queries = [[1, 2, 100], [2, 5, 100], [3, 4, 100]]
2 5 100
3 4 100
```

Sample Output

```
200
```

Explanation

After the first update the list is `100 100 0 0 0`.

After the second update list is `100 200 100 100 100`.

After the third update list is `100 200 200 200 100`.

The maximum value is 200.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Sample Test case 2

Input (stdin) Download

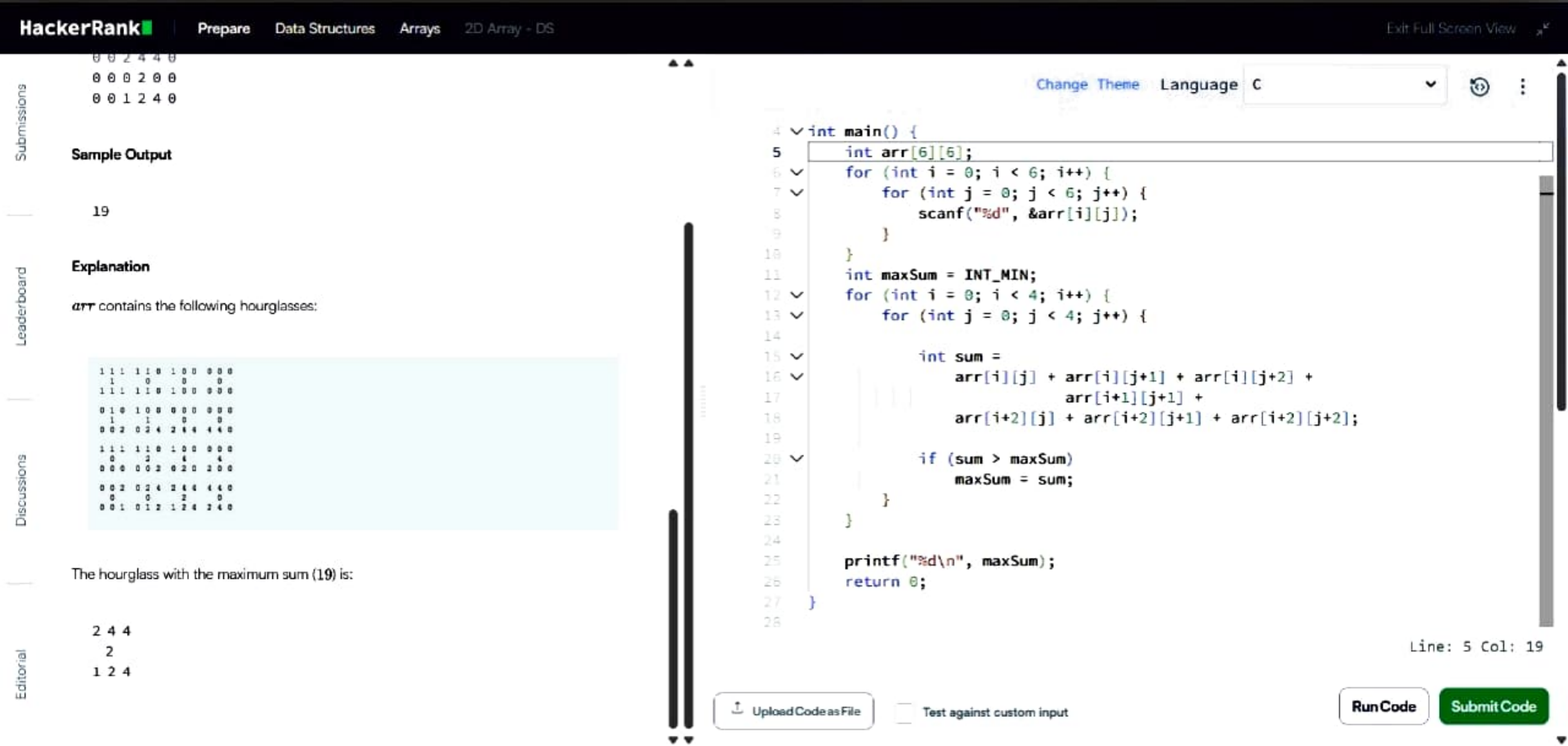
```
5 3
1 2 100
2 5 100
3 4 100
```

Your Output (stdout)

```
200
```

Expected Output Download

```
200
```



```
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:

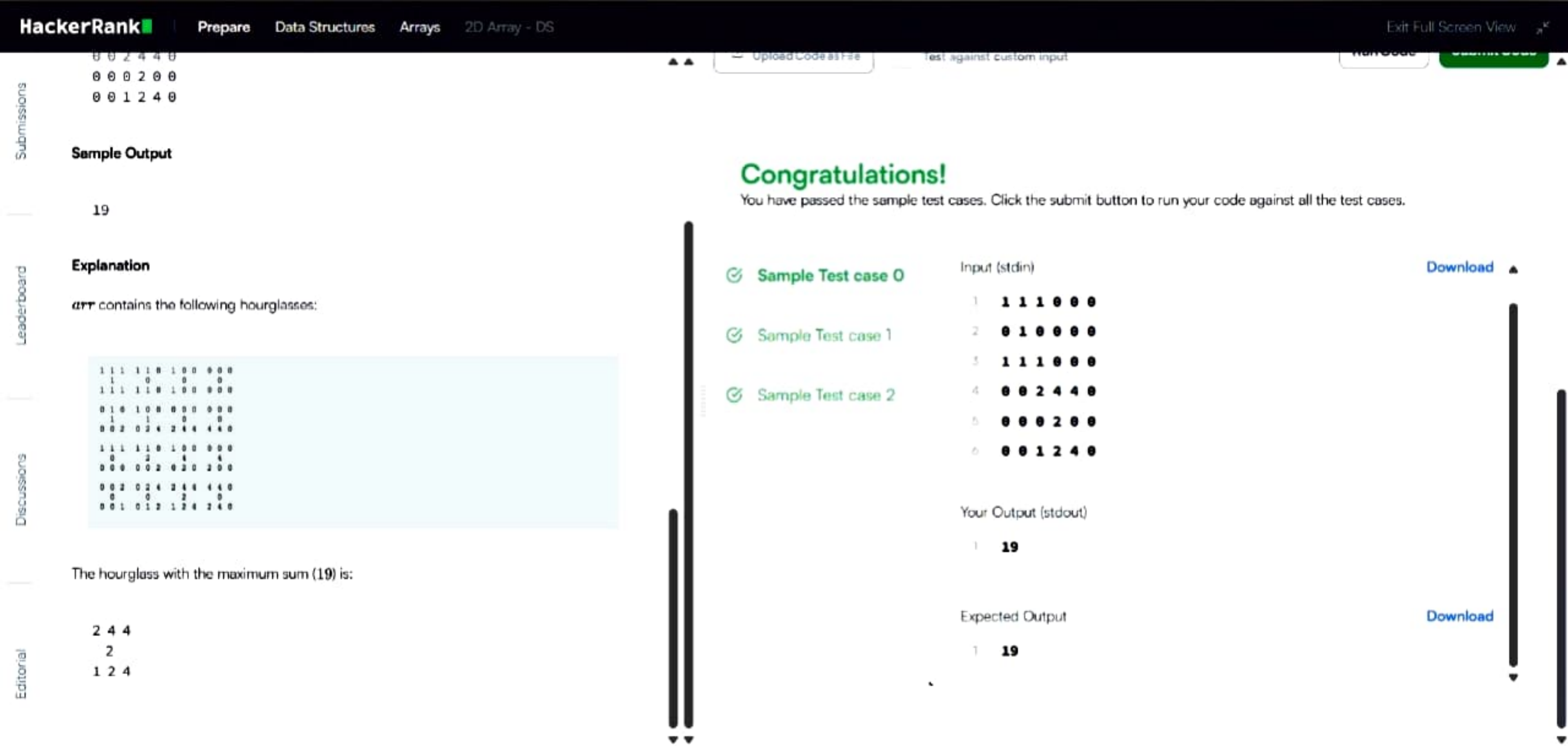
```
1 1 1   1 1 0   1 0 0   0 0 0
  1       0       0       0
1 1 1   1 1 0   1 0 0   0 0 0

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
```

```c
int main() {
    int arr[6][6];
    for (int i = 0; i < 6; i++) {
        for (int j = 0; j < 6; j++) {
            scanf("%d", &arr[i][j]);
        }
    }
    int maxSum = INT_MIN;
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {

            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];

            if (sum > maxSum)
                maxSum = sum;
        }
    }

    printf("%d\n", maxSum);
    return 0;
}
```

Line: 5 Col: 19



HackerRank | Prepare | Data Structures | Arrays | 2D Array - DS | Exit Full Screen View

Submissions | Leaderboard | Discussions | Editorial

```
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
```

Upload Code as File | Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0
✓ Sample Test case 1
✓ Sample Test case 2

Input (stdin) Download

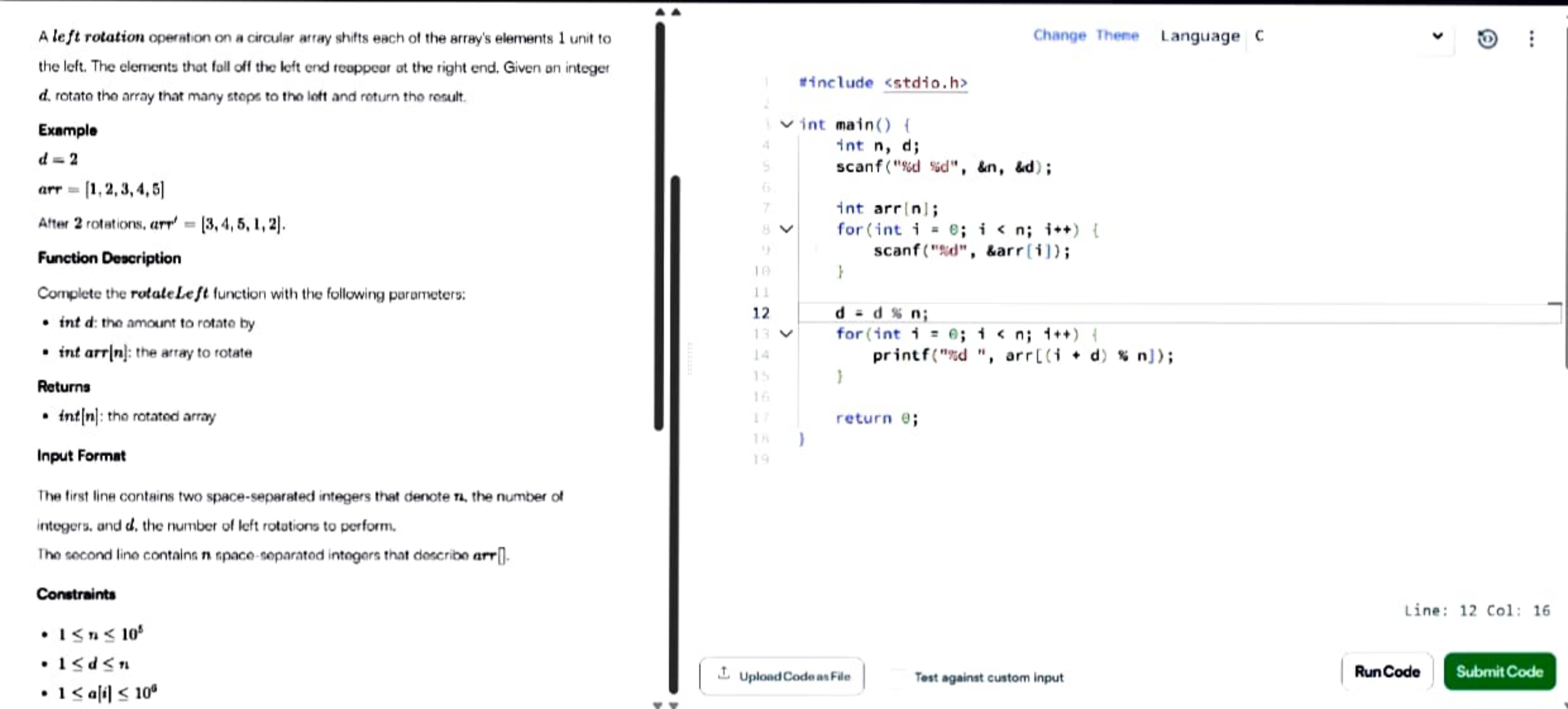
```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 2 4 4 0
0 0 0 2 0 0
0 0 1 2 4 0
```

Your Output (stdout)

```
19
```

Expected Output Download

```
19
```


A *left rotation* operation on a circular array shifts each of the array's elements 1 unit to the left. The elements that fall off the left end reappear at the right end. Given an integer d, rotate the array that many steps to the 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 with 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.

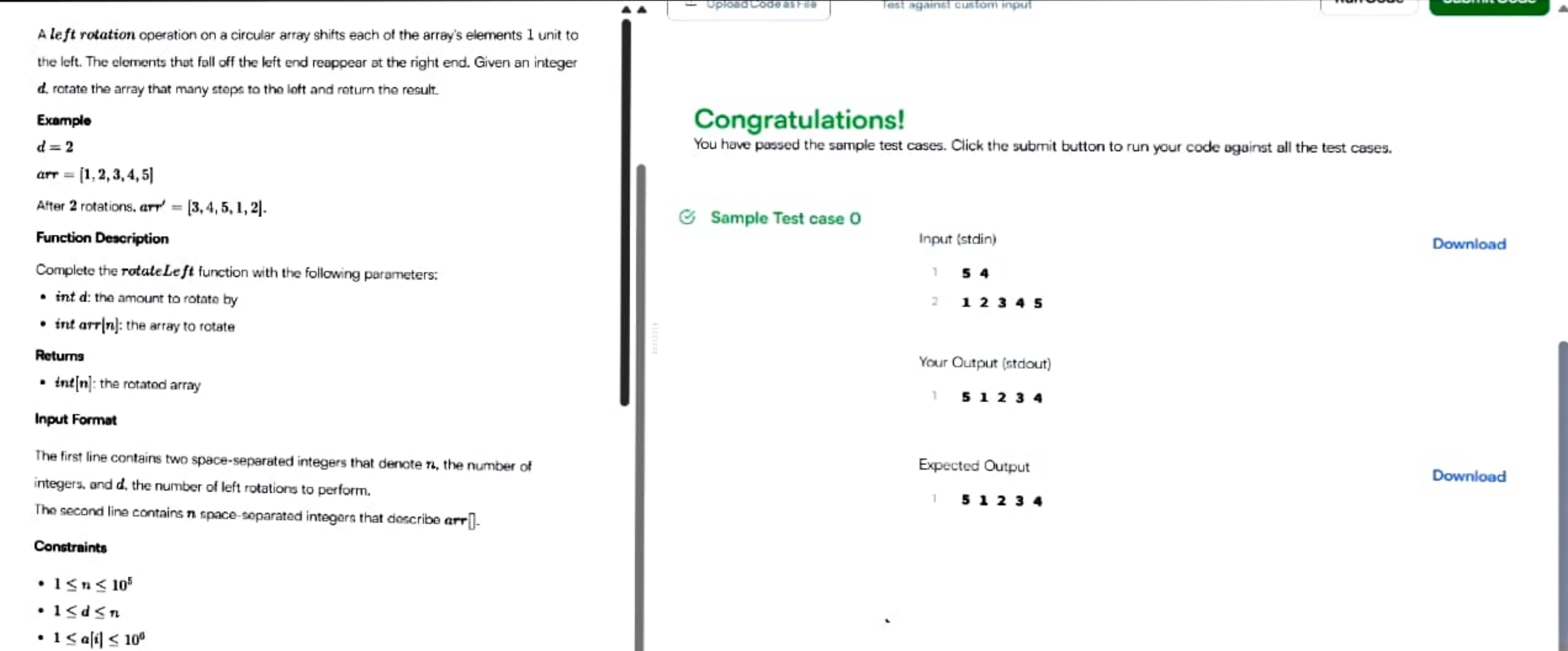
The second line contains n space-separated integers that describe $arr[]$.

Constraints

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

```c
#include <stdio.h>

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

    int arr[n];
    for(int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    d = d % n;
    for(int i = 0; i < n; i++) {
        printf("%d ", arr[(i + d) % n]);
    }

    return 0;
}
```

```
Line: 12 Col: 16
```



A *left rotation* operation on a circular array shifts each of the array's elements 1 unit to the left. The elements that fall off the left end reappear at the right end. Given an integer d, rotate the array that many steps to the 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 with 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

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

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin)

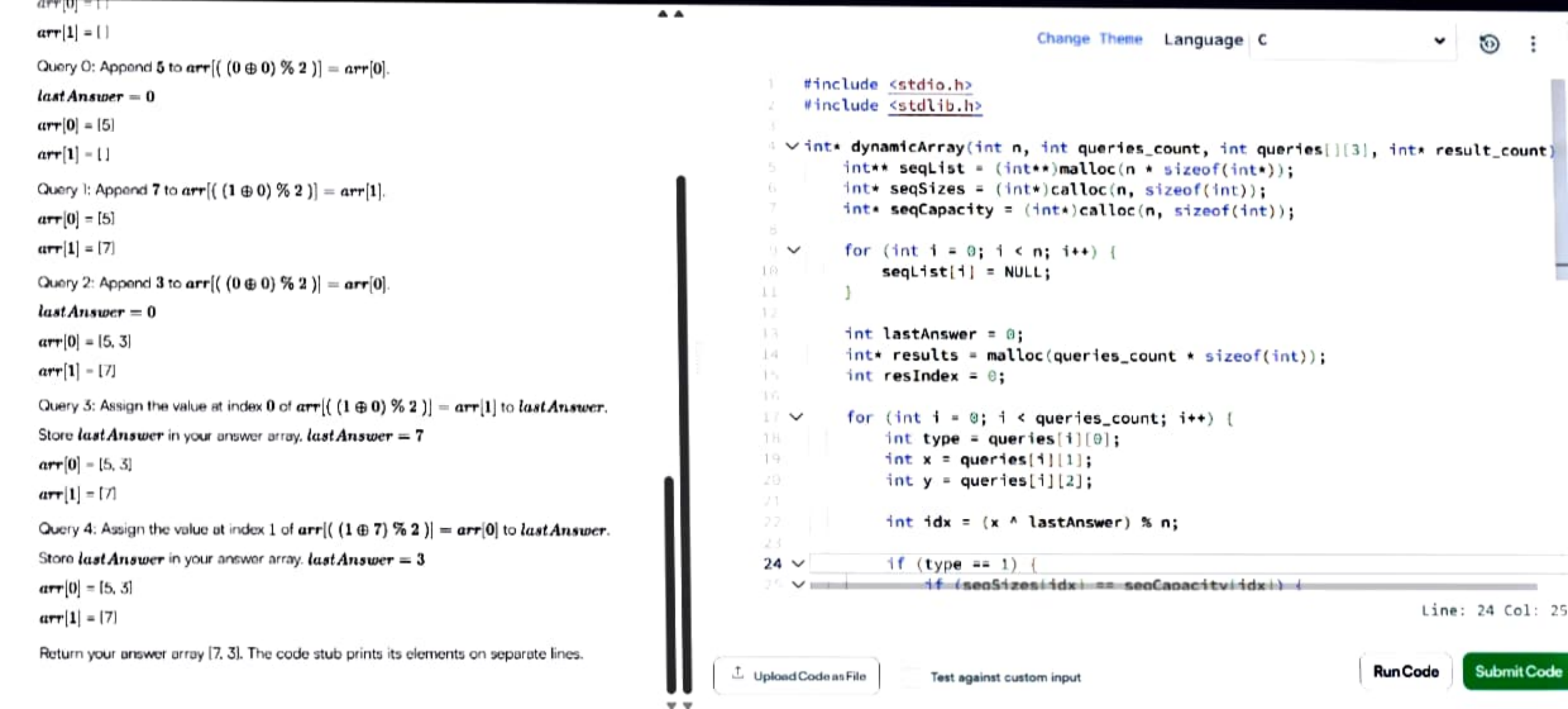
```
5 4
1 2 3 4 5
```

Your Output (stdout)

```
5 1 2 3 4
```

Expected Output

```
5 1 2 3 4
```


$arr[0] = []$

$arr[1] = []$

Query 0: Append 5 to $arr[(\ (0 \oplus 0)\ \%\ 2\ )] = arr[0]$.

$lastAnswer = 0$

$arr[0]$ = [5]

$arr[1]$ = []

Query 1: Append 7 to $arr[(\ (1 \oplus 0)\ \%\ 2\ )] = arr[1]$.

$arr[0]$ = [5]

$arr[1]$ = [7]

Query 2: Append 3 to $arr[(\ (0 \oplus 0)\ \%\ 2\ )] = arr[0]$.

$lastAnswer = 0$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Query 3: Assign the value at index 0 of $arr[(\ (1 \oplus 0)\ \%\ 2\ )] = arr[1]$ to $lastAnswer$.

Store $lastAnswer$ in your answer array. $lastAnswer = 7$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Query 4: Assign the value at index 1 of $arr[(\ (1 \oplus 7)\ \%\ 2\ )] = arr[0]$ to $lastAnswer$.

Store $lastAnswer$ in your answer array. $lastAnswer = 3$

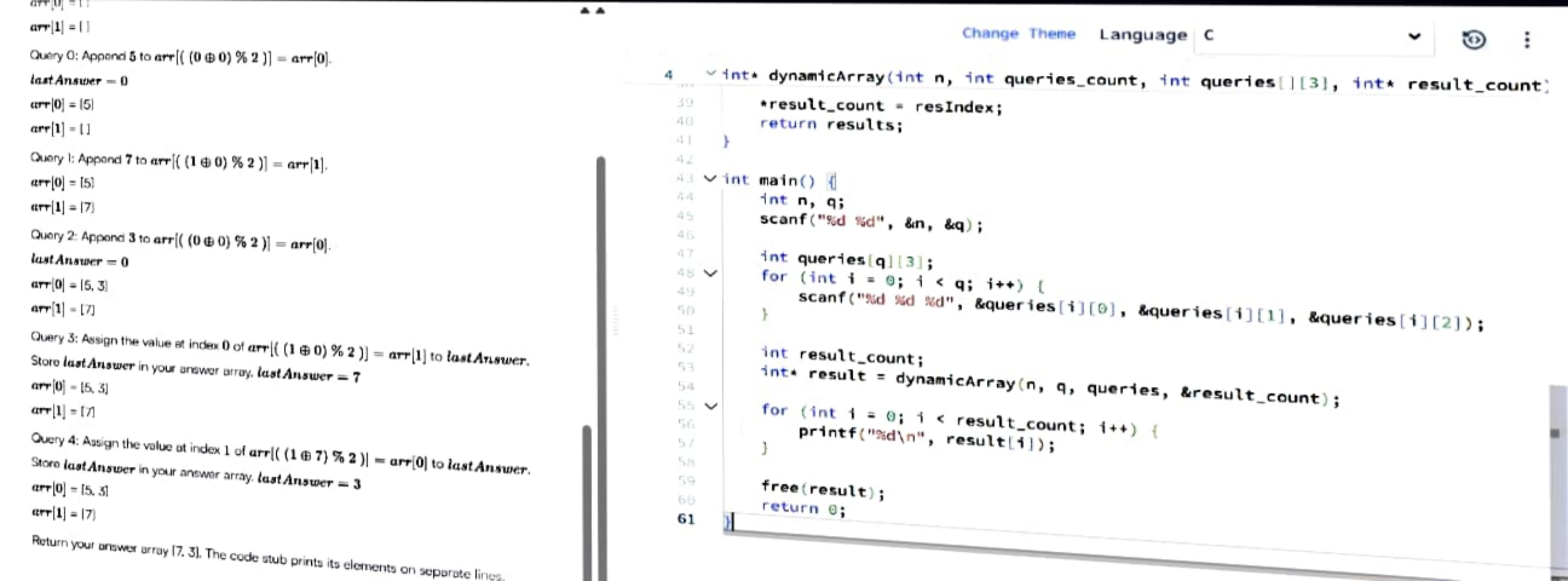
$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Return your answer array [7, 3]. The code stub prints its elements on separate lines.

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

int* dynamicArray(int n, int queries_count, int queries[][3], int* result_count)
    int** seqList = (int**)malloc(n * sizeof(int*));
    int* seqSizes = (int*)calloc(n, sizeof(int));
    int* seqCapacity = (int*)calloc(n, sizeof(int));

    for (int i = 0; i < n; i++) {
        seqList[i] = NULL;
    }

    int lastAnswer = 0;
    int* results = malloc(queries_count * sizeof(int));
    int resIndex = 0;

    for (int i = 0; i < queries_count; i++) {
        int type = queries[i][0];
        int x = queries[i][1];
        int y = queries[i][2];

        int idx = (x ^ lastAnswer) % n;

        if (type == 1) {
            if (seqSizes[idx] == seqCapacity[idx]) {
```

Line: 24 Col: 25


$arr[1]$ = []

Query 0: Append 5 to $arr[(\,(0 \oplus 0)\ \%\ 2\,)] = arr[0]$.

$lastAnswer = 0$

$arr[0]$ = [5]

$arr[1]$ = []

Query 1: Append 7 to $arr[(\,(1 \oplus 0)\ \%\ 2\,)] = arr[1]$.

$arr[0]$ = [5]

$arr[1]$ = [7]

Query 2: Append 3 to $arr[(\,(0 \oplus 0)\ \%\ 2\,)] = arr[0]$.

$lastAnswer = 0$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Query 3: Assign the value at index 0 of $arr[(\,(1 \oplus 0)\ \%\ 2\,)] = arr[1]$ to $lastAnswer$.
Store $lastAnswer$ in your answer array. $lastAnswer = 7$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Query 4: Assign the value at index 1 of $arr[(\,(1 \oplus 7)\ \%\ 2\,)] = arr[0]$ to $lastAnswer$.
Store $lastAnswer$ in your answer array. $lastAnswer = 3$

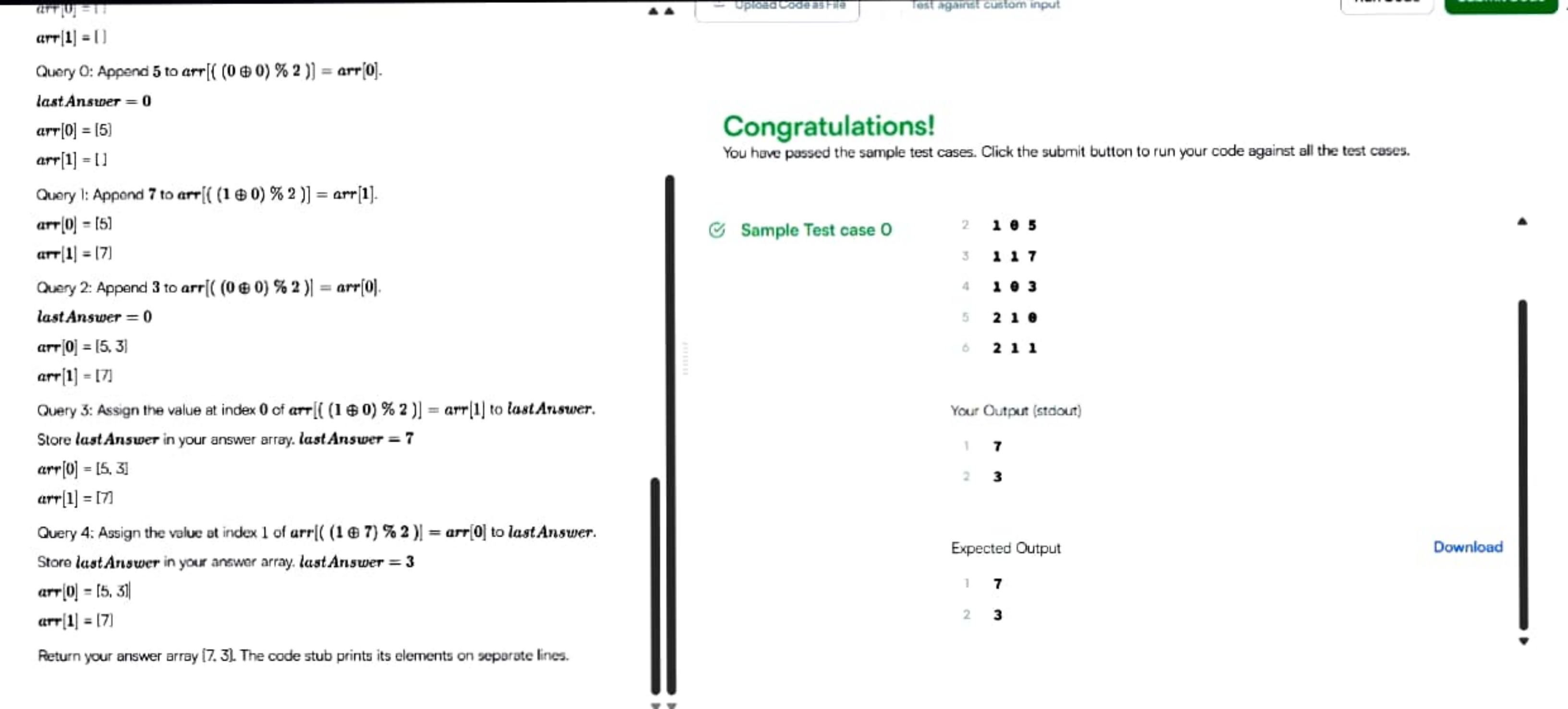
$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Return your answer array [7, 3]. The code stub prints its elements on separate lines.

Change Theme Language C

```c
int* dynamicArray(int n, int queries_count, int queries[][3], int* result_count)
    *result_count = resIndex;
    return results;
}

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

    int queries[q][3];
    for (int i = 0; i < q; i++) {
        scanf("%d %d %d", &queries[i][0], &queries[i][1], &queries[i][2]);
    }

    int result_count;
    int* result = dynamicArray(n, q, queries, &result_count);
    for (int i = 0; i < result_count; i++) {
        printf("%d\n", result[i]);
    }

    free(result);
    return 0;
}
```



$arr[0]$ = [ ]

$arr[1]$ = [ ]

Query 0: Append 5 to $arr[(\ (0 \oplus 0)\ \%\ 2\ )] = arr[0]$.

$lastAnswer = 0$

$arr[0]$ = [5]

$arr[1]$ = [ ]

Query 1: Append 7 to $arr[(\ (1 \oplus 0)\ \%\ 2\ )] = arr[1]$.

$arr[0]$ = [5]

$arr[1]$ = [7]

Query 2: Append 3 to $arr[(\ (0 \oplus 0)\ \%\ 2\ )] = arr[0]$.

$lastAnswer = 0$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Query 3: Assign the value at index 0 of $arr[(\ (1 \oplus 0)\ \%\ 2\ )] = arr[1]$ to $lastAnswer$.

Store $lastAnswer$ in your answer array. $lastAnswer = 7$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Query 4: Assign the value at index 1 of $arr[(\ (1 \oplus 7)\ \%\ 2\ )] = arr[0]$ to $lastAnswer$.

Store $lastAnswer$ in your answer array. $lastAnswer = 3$

$arr[0]$ = [5, 3]

$arr[1]$ = [7]

Return your answer array [7, 3]. The code stub prints its elements on separate lines.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

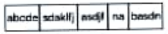
```
1 0 5
1 1 7
1 0 3
2 1 0
2 1 1
```

Your Output (stdout)

```
7
3
```

Expected Output

```
7
3
```



Array: queries

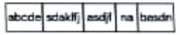
```
abcde
sdaklfj
asdjf
na
basdn
sdaklfj
asdjf
na
asdjf
na
basdn
sdaklfj
asdjf
5
abcde
sdaklfj
asdjf
na
basdn
```

Sample Output 3

```
1
3
4
3
2
```

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

#define MAX 1000
#define MAX_LEN 1000

int main() {
    int n, q;
    char strings[MAX][MAX_LEN];
    char queries[MAX][MAX_LEN];
    int counts[MAX] = {0};

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%s", strings[i]);
    }

    scanf("%d", &q);

    for (int i = 0; i < q; i++) {
        scanf("%s", queries[i]);
    }
    for (int i = 0; i < q; i++) {
        int count = 0:
```

Line: 7 Col: 13



Array: queries

Sample Output 3

```
1
3
4
3
2
```

```
abcde
sdaklfj
asdjf
na
besdn
sdaklfj
asdjf
na
asdjf
na
besdn
sdaklfj
asdjf
5
abcde
sdaklfj
asdjf
na
besdn
```

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Sample Test case 2

```
aba
xzxb
ab
```

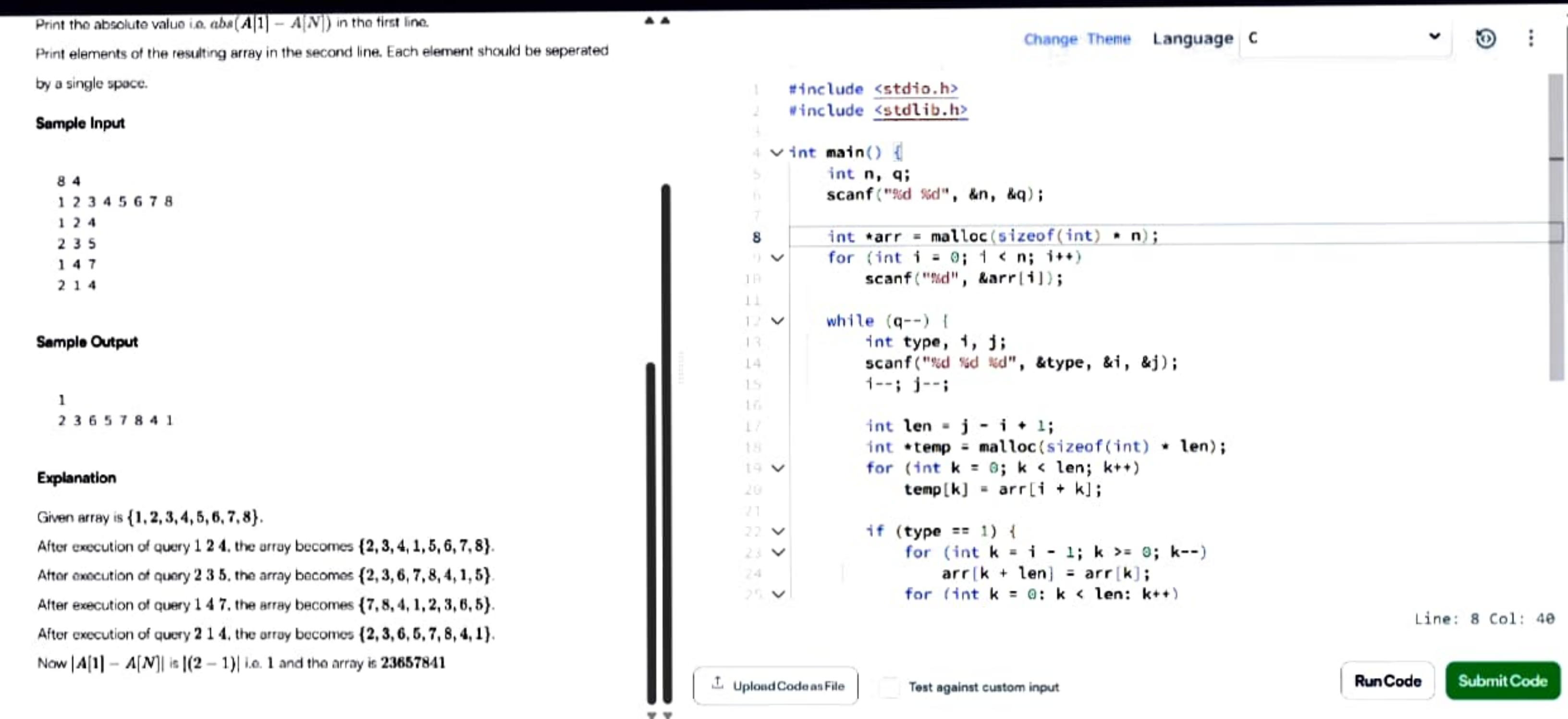
Your Output (stdout)

```
2
1
0
```

Expected Output

Down

```
2
1
0
```



Print the absolute value i.e. $abs(A[1] - A[N])$ in the first line.

Print elements of the resulting array in the second line. Each element should be seperated by a single space.

Sample Input

```
8 4
1 2 3 4 5 6 7 8
1 2 4
2 3 5
1 4 7
2 1 4
```

Sample Output

```
1
2 3 6 5 7 8 4 1
```

Explanation

Given array is $\{1, 2, 3, 4, 5, 6, 7, 8\}$.

After execution of query 1 2 4, the array becomes $\{2, 3, 4, 1, 5, 6, 7, 8\}$.

After execution of query 2 3 5, the array becomes $\{2, 3, 6, 7, 8, 4, 1, 5\}$.

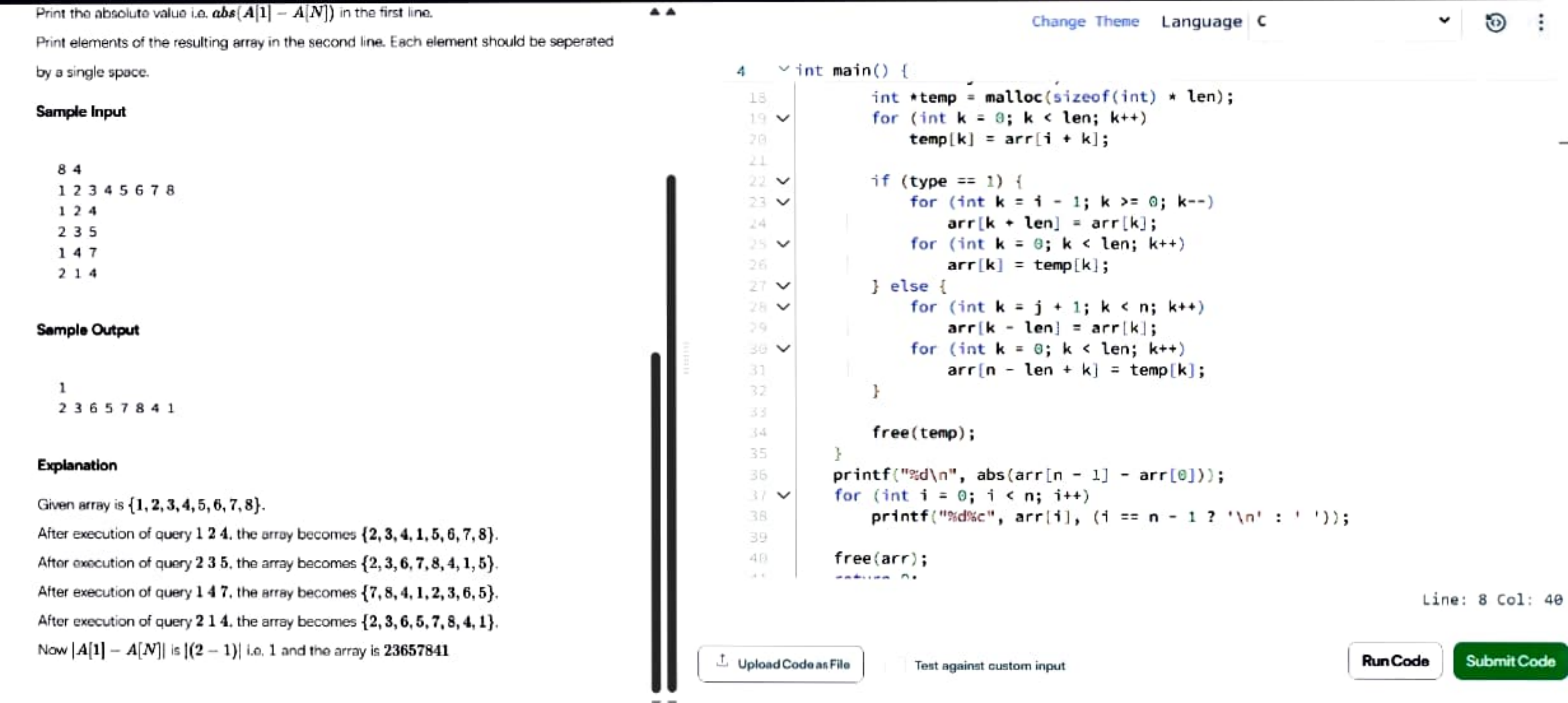
After execution of query 1 4 7, the array becomes $\{7, 8, 4, 1, 2, 3, 6, 5\}$.

After execution of query 2 1 4, the array becomes $\{2, 3, 6, 5, 7, 8, 4, 1\}$.

Now $|A[1] - A[N]|$ is $|(2 - 1)|$ i.e. 1 and the array is 23657841

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

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

    int *arr = malloc(sizeof(int) * n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    while (q--) {
        int type, i, j;
        scanf("%d %d %d", &type, &i, &j);
        i--; j--;

        int len = j - i + 1;
        int *temp = malloc(sizeof(int) * len);
        for (int k = 0; k < len; k++)
            temp[k] = arr[i + k];

        if (type == 1) {
            for (int k = i - 1; k >= 0; k--)
                arr[k + len] = arr[k];
            for (int k = 0; k < len; k++)
```

Line: 8 Col: 40



Print the absolute value i.e. $abs(A[1] - A[N])$ in the first line.

Print elements of the resulting array in the second line. Each element should be seperated by a single space.

Sample Input

```
8 4
1 2 3 4 5 6 7 8
1 2 4
2 3 5
1 4 7
2 1 4
```

Sample Output

```
1
2 3 6 5 7 8 4 1
```

Explanation

Given array is $\{1,2,3,4,5,6,7,8\}$.

After execution of query $1\ 2\ 4$, the array becomes $\{2,3,4,1,5,6,7,8\}$.

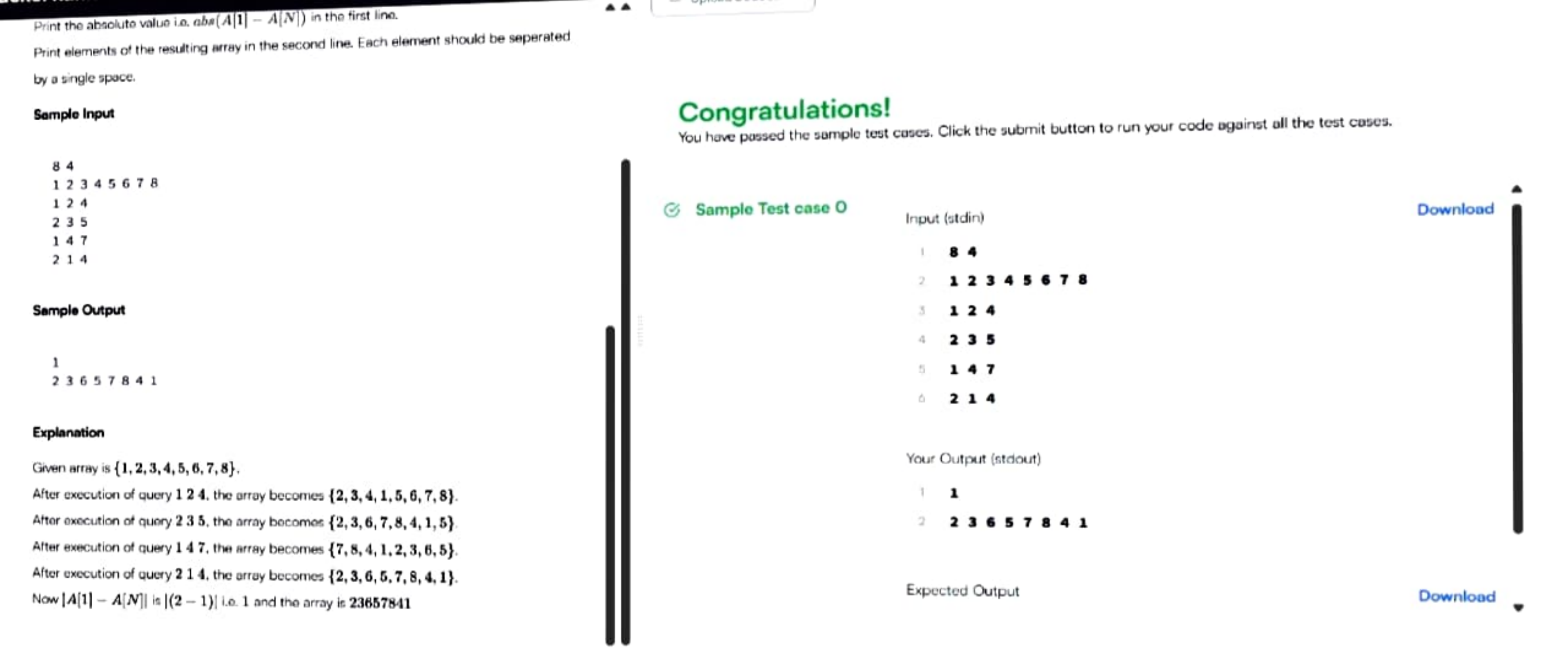
After execution of query $2\ 3\ 5$, the array becomes $\{2,3,6,7,8,4,1,5\}$.

After execution of query $1\ 4\ 7$, the array becomes $\{7,8,4,1,2,3,6,5\}$.

After execution of query $2\ 1\ 4$, the array becomes $\{2,3,6,5,7,8,4,1\}$.

Now $|A[1] - A[N]|$ is $|(2-1)|$ i.e. 1 and the array is 23657841

```
int main() {
        int *temp = malloc(sizeof(int) * len);
        for (int k = 0; k < len; k++)
            temp[k] = arr[i + k];

        if (type == 1) {
            for (int k = i - 1; k >= 0; k--)
                arr[k + len] = arr[k];
            for (int k = 0; k < len; k++)
                arr[k] = temp[k];
        } else {
            for (int k = j + 1; k < n; k++)
                arr[k - len] = arr[k];
            for (int k = 0; k < len; k++)
                arr[n - len + k] = temp[k];
        }

        free(temp);
    }
    printf("%d\n", abs(arr[n - 1] - arr[0]));
    for (int i = 0; i < n; i++)
        printf("%d%c", arr[i], (i == n - 1 ? '\n' : ' '));

    free(arr);
```



Print the absolute value i.e. $abs(A[1] - A[N])$ in the first line.

Print elements of the resulting array in the second line. Each element should be seperated by a single space.

Sample Input

```
8 4
1 2 3 4 5 6 7 8
1 2 4
2 3 5
1 4 7
2 1 4
```

Sample Output

```
1
2 3 6 5 7 8 4 1
```

Explanation

Given array is $\{1,2,3,4,5,6,7,8\}$.

After execution of query $1\ 2\ 4$, the array becomes $\{2,3,4,1,5,6,7,8\}$.

After execution of query $2\ 3\ 5$, the array becomes $\{2,3,6,7,8,4,1,5\}$.

After execution of query $1\ 4\ 7$, the array becomes $\{7,8,4,1,2,3,6,5\}$.

After execution of query $2\ 1\ 4$, the array becomes $\{2,3,6,5,7,8,4,1\}$.

Now $|A[1] - A[N]|$ is $|(2-1)|$ i.e. 1 and the array is 23657841

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

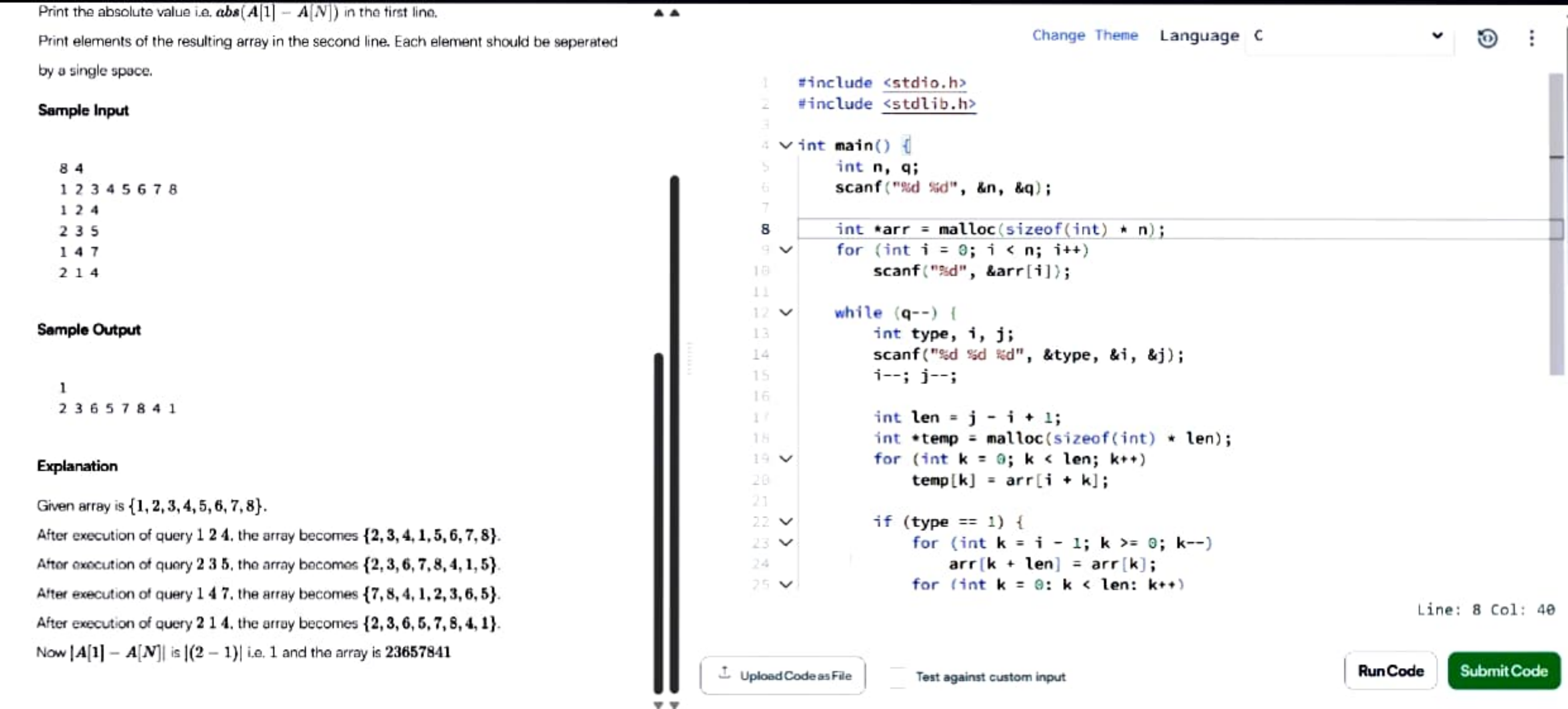
Input (stdin) Download

```
8 4
1 2 3 4 5 6 7 8
1 2 4
2 3 5
1 4 7
2 1 4
```

Your Output (stdout)

```
1
2 3 6 5 7 8 4 1
```

Expected Output Download



Print the absolute value i.e. $abs(A[1] - A[N])$ in the first line.

Print elements of the resulting array in the second line. Each element should be seperated by a single space.

Sample Input

```
8 4
1 2 3 4 5 6 7 8
1 2 4
2 3 5
1 4 7
2 1 4
```

Sample Output

```
1
2 3 6 5 7 8 4 1
```

Explanation

Given array is $\{1, 2, 3, 4, 5, 6, 7, 8\}$.

After execution of query 1 2 4, the array becomes $\{2, 3, 4, 1, 5, 6, 7, 8\}$.

After execution of query 2 3 5, the array becomes $\{2, 3, 6, 7, 8, 4, 1, 5\}$.

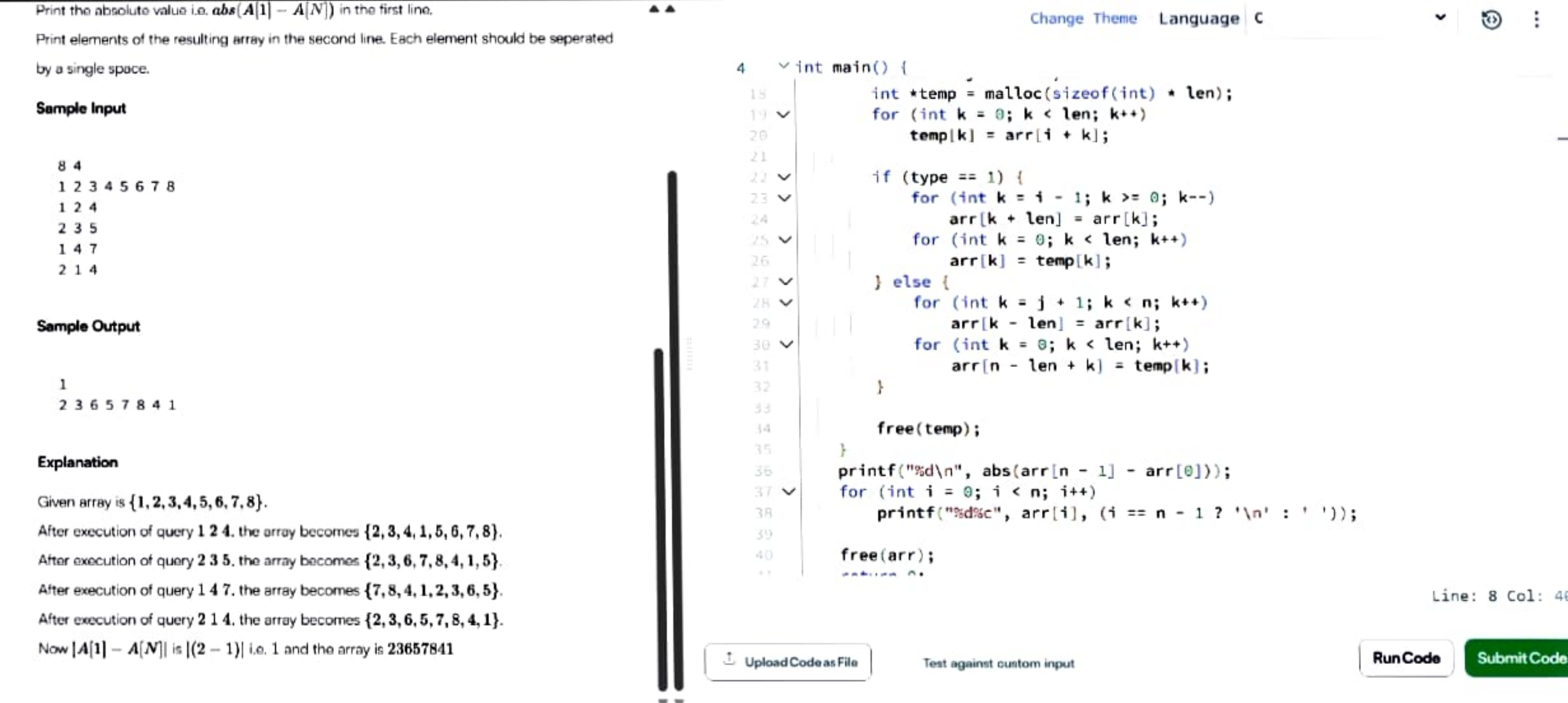
After execution of query 1 4 7, the array becomes $\{7, 8, 4, 1, 2, 3, 6, 5\}$.

After execution of query 2 1 4, the array becomes $\{2, 3, 6, 5, 7, 8, 4, 1\}$.

Now $|A[1] - A[N]|$ is $|(2 - 1)|$ i.e. 1 and the array is 23657841

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

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

    int *arr = malloc(sizeof(int) * n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    while (q--) {
        int type, i, j;
        scanf("%d %d %d", &type, &i, &j);
        i--; j--;

        int len = j - i + 1;
        int *temp = malloc(sizeof(int) * len);
        for (int k = 0; k < len; k++)
            temp[k] = arr[i + k];

        if (type == 1) {
            for (int k = i - 1; k >= 0; k--)
                arr[k + len] = arr[k];
            for (int k = 0; k < len; k++)
```

Line: 8 Col: 40


Print the absolute value i.e. $abs(A[1] - A[N])$ in the first line.

Print elements of the resulting array in the second line. Each element should be seperated by a single space.

Sample Input

```
8 4
1 2 3 4 5 6 7 8
1 2 4
2 3 5
1 4 7
2 1 4
```

Sample Output

```
1
2 3 6 5 7 8 4 1
```

Explanation

Given array is $\{1, 2, 3, 4, 5, 6, 7, 8\}$.

After execution of query **1 2 4**, the array becomes $\{2, 3, 4, 1, 5, 6, 7, 8\}$.

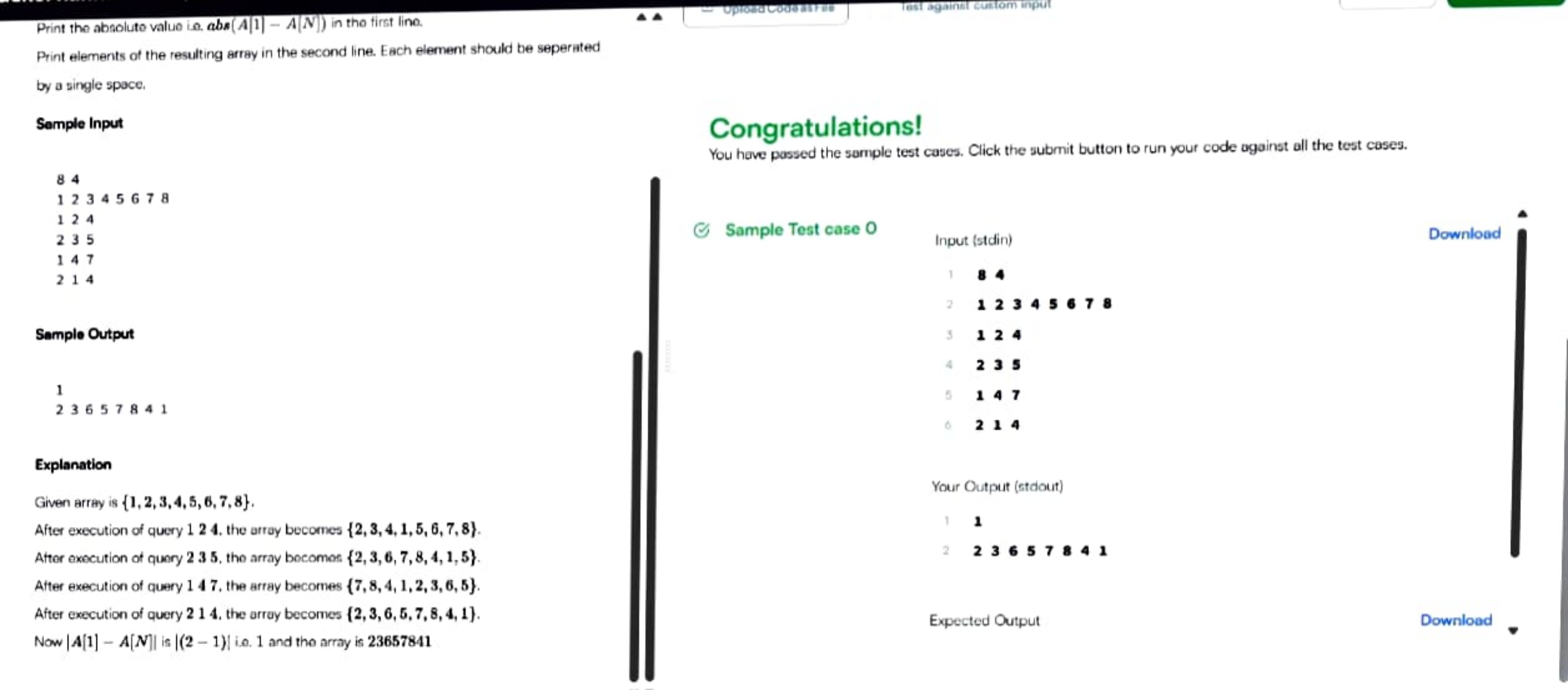
After execution of query **2 3 5**, the array becomes $\{2, 3, 6, 7, 8, 4, 1, 5\}$.

After execution of query **1 4 7**, the array becomes $\{7, 8, 4, 1, 2, 3, 6, 5\}$.

After execution of query **2 1 4**, the array becomes $\{2, 3, 6, 5, 7, 8, 4, 1\}$.

Now $|A[1] - A[N]|$ is $|(2 - 1)|$ i.e. 1 and the array is **23657841**

```c
int main() {
    int *temp = malloc(sizeof(int) * len);
    for (int k = 0; k < len; k++)
      temp[k] = arr[i + k];

    if (type == 1) {
      for (int k = i - 1; k >= 0; k--)
        arr[k + len] = arr[k];
      for (int k = 0; k < len; k++)
        arr[k] = temp[k];
    } else {
      for (int k = j + 1; k < n; k++)
        arr[k - len] = arr[k];
      for (int k = 0; k < len; k++)
        arr[n - len + k] = temp[k];
    }

    free(temp);
  }
  printf("%d\n", abs(arr[n - 1] - arr[0]));
  for (int i = 0; i < n; i++)
    printf("%d%c", arr[i], (i == n - 1 ? '\n' : ' '));

  free(arr);
  return 0;
```



Print the absolute value i.e. $abs(A[1] - A[N])$ in the first line.

Print elements of the resulting array in the second line. Each element should be seperated by a single space.

Sample Input

```
8 4
1 2 3 4 5 6 7 8
1 2 4
2 3 5
1 4 7
2 1 4
```

Sample Output

```
1
2 3 6 5 7 8 4 1
```

Explanation

Given array is $\{1, 2, 3, 4, 5, 6, 7, 8\}$.

After execution of query 1 2 4, the array becomes $\{2, 3, 4, 1, 5, 6, 7, 8\}$.

After execution of query 2 3 5, the array becomes $\{2, 3, 6, 7, 8, 4, 1, 5\}$.

After execution of query 1 4 7, the array becomes $\{7, 8, 4, 1, 2, 3, 6, 5\}$.

After execution of query 2 1 4, the array becomes $\{2, 3, 6, 5, 7, 8, 4, 1\}$.

Now $|A[1] - A[N]|$ is $|(2 - 1)|$ i.e. 1 and the array is 23657841

Upload Code as File

Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
8 4
1 2 3 4 5 6 7 8
1 2 4
2 3 5
1 4 7
2 1 4
```

Your Output (stdout)

```
1
2 3 6 5 7 8 4 1
```

Expected Output Download

abcde
sdaklfj
asdjf
na
basdn
Array: queries

```
abcde
sdaklfj
asdjf
na
basdn
sdaklfj
asdjf
na
asdjf
na
basdn
sdaklfj
asdjf
5
abcde
sdaklfj
asdjf
na
basdn
```

Sample Output 3

```
1
3
4
3
2
```

Change Theme Language C

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

#define MAX 1000
#define MAX_LEN 1000

int main() {
    int n, q;
    char strings[MAX][MAX_LEN];
    char queries[MAX][MAX_LEN];
    int counts[MAX] = {0};

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%s", strings[i]);
    }

    scanf("%d", &q);

    for (int i = 0; i < q; i++) {
        scanf("%s", queries[i]);
    }
    for (int i = 0; i < q; i++) {
        int count = 0;
```

Line: 7 Col: 13

Upload Code as File Test against custom input Run Code Submit Code

```
abcde
sdaklfj
asdjf
na
basdn
sdaklfj
asdjf
na
asdjf
na
basdn
sdaklfj
asdjf
5
abcde
sdaklfj
asdjf
na
basdn
```

abcde
sdaklfj
asdjf
na
basdn

Array: queries

Sample Output 3

```
1
3
4
3
2
```

Upload Code as File
Test against custom input
Run Code
Submit Cod

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Sample Test case 2

```
aba
xzxb
ab
```

Your Output (stdout)

```
2
1
0
```

Expected Output

Download

```
2
1
0
```