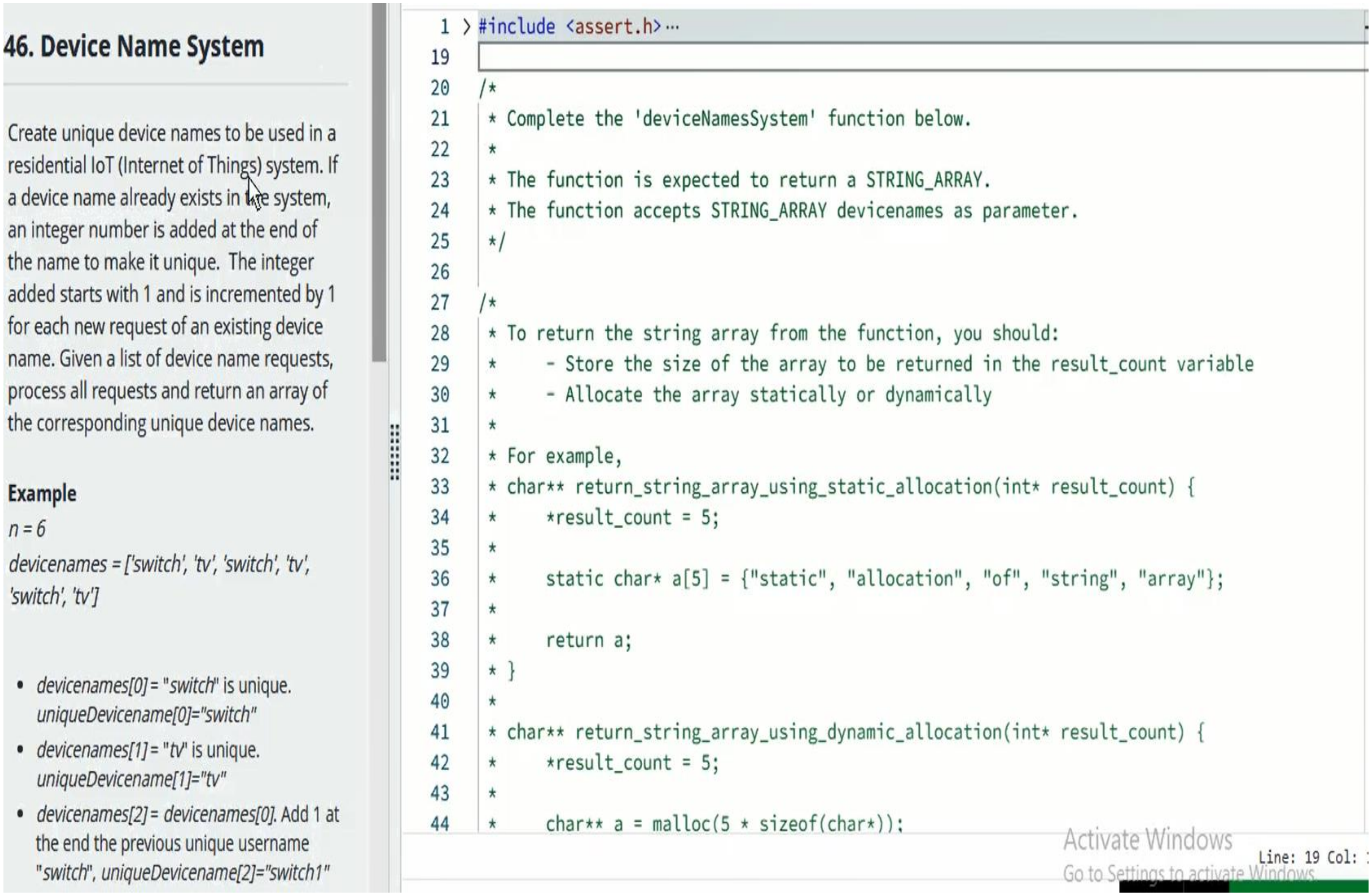


46. Device Name System

Create unique device names to be used in a residential IoT (Internet of Things) system. If a device name already exists in the system, an integer number is added at the end of the name to make it unique. The integer added starts with 1 and is incremented by 1 for each new request of an existing device name. Given a list of device name requests, process all requests and return an array of the corresponding unique device names.

Example

n = 6

devicenames = ['switch', 'tv', 'switch', 'tv', 'switch', 'tv']

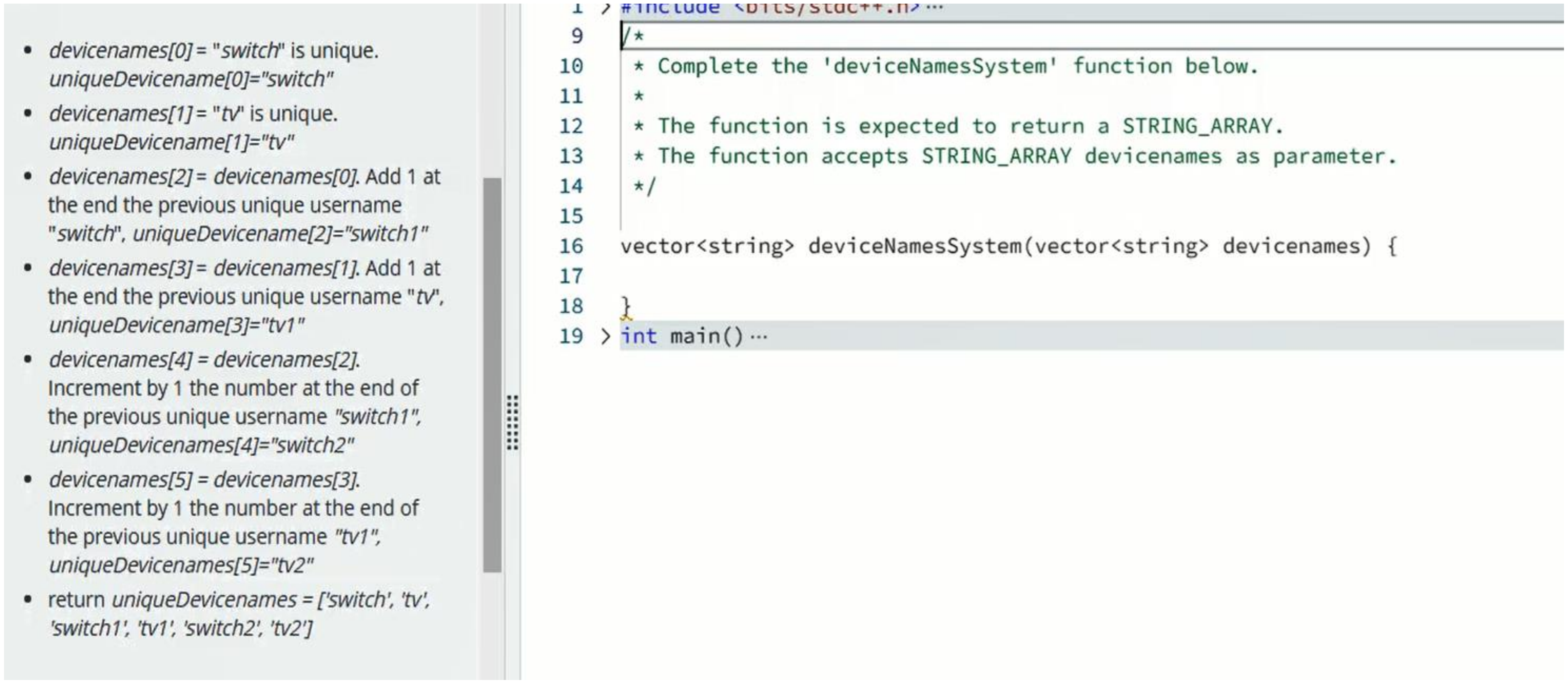
- *devicenames[0]* = "*switch*" is unique. *uniqueDevicename[0]="switch"*
- *devicenames[1]* = "*tv*" is unique. *uniqueDevicename[1]="tv"*
- *devicenames[2]* = *devicenames[0]*. Add 1 at the end the previous unique username "*switch*", *uniqueDevicename[2]="switch1"*

```
#include <assert.h>···

/*
 * Complete the 'deviceNamesSystem' function below.
 *
 * The function is expected to return a STRING_ARRAY.
 * The function accepts STRING_ARRAY devicenames as parameter.
 */

/*
 * To return the string array from the function, you should:
 *     - Store the size of the array to be returned in the result_count variable
 *     - Allocate the array statically or dynamically
 *
 * For example,
 * char** return_string_array_using_static_allocation(int* result_count) {
 *     *result_count = 5;
 *
 *     static char* a[5] = {"static", "allocation", "of", "string", "array"};
 *
 *     return a;
 * }
 *
 * char** return_string_array_using_dynamic_allocation(int* result_count) {
 *     *result_count = 5;
 *
 *     char** a = malloc(5 * sizeof(char*));
```

Line: 19 Col:

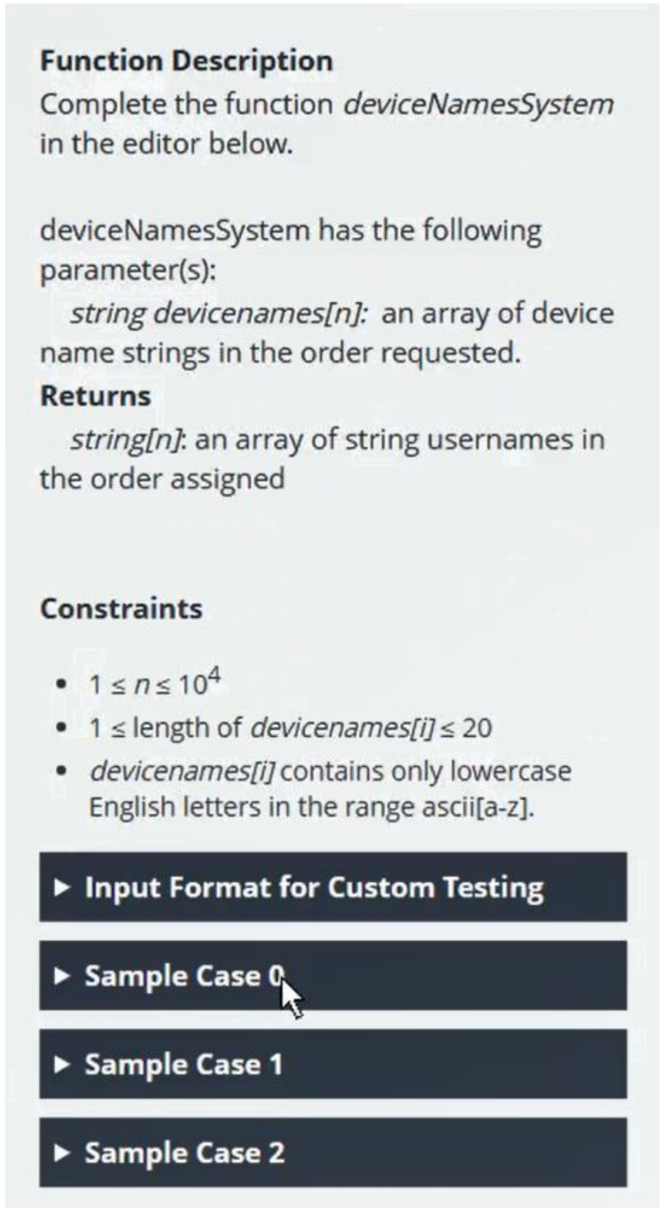


- *devicenames[0]* = "*switch*" is unique. *uniqueDevicename[0]="switch"*
- *devicenames[1]* = "*tv*" is unique. *uniqueDevicename[1]="tv"*
- *devicenames[2]* = *devicenames[0]*. Add 1 at the end the previous unique username "*switch*", *uniqueDevicename[2]="switch1"*
- *devicenames[3]* = *devicenames[1]*. Add 1 at the end the previous unique username "*tv*", *uniqueDevicename[3]="tv1"*
- *devicenames[4]* = *devicenames[2]*. Increment by 1 the number at the end of the previous unique username *"switch1"*, *uniqueDevicenames[4]="switch2"*
- *devicenames[5]* = *devicenames[3]*. Increment by 1 the number at the end of the previous unique username *"tv1"*, *uniqueDevicenames[5]="tv2"*
- return *uniqueDevicenames = ['switch', 'tv', 'switch1', 'tv1', 'switch2', 'tv2']*

```
#include <bits/stdc++.h> ···
/*
 * Complete the 'deviceNamesSystem' function below.
 *
 * The function is expected to return a STRING_ARRAY.
 * The function accepts STRING_ARRAY devicenames as parameter.
 */

vector<string> deviceNamesSystem(vector<string> devicenames) {

}
int main() ···
```



Function Description

Complete the function *deviceNamesSystem* in the editor below.

deviceNamesSystem has the following parameter(s):

string devicenames[n]: an array of device name strings in the order requested.

Returns

string[n]: an array of string usernames in the order assigned

Constraints

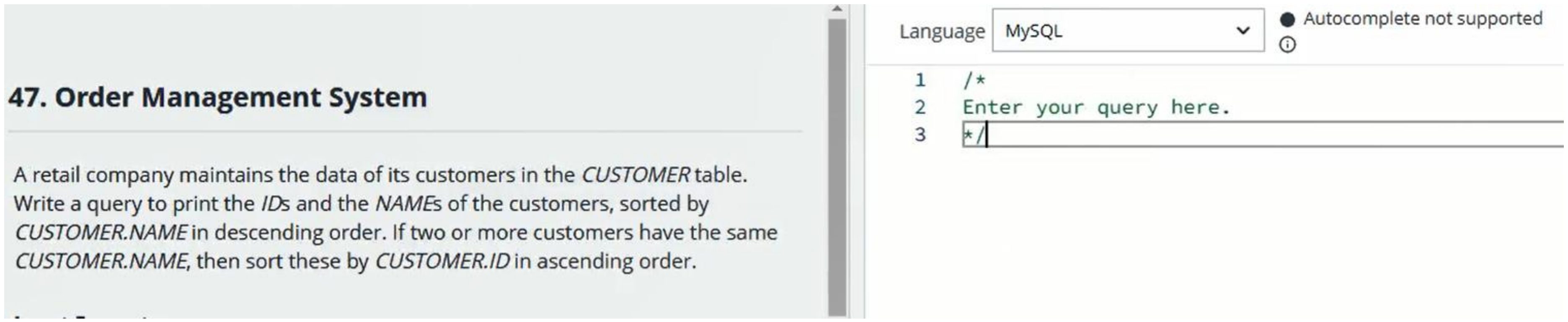
- $1 \le n \le 10^4$
- $1 \le$ length of *devicenames[i]* ≤ 20
- *devicenames[i]* contains only lowercase English letters in the range ascii[a-z].

▶ **Input Format for Custom Testing**

▶ **Sample Case 0**

▶ **Sample Case 1**

▶ **Sample Case 2**



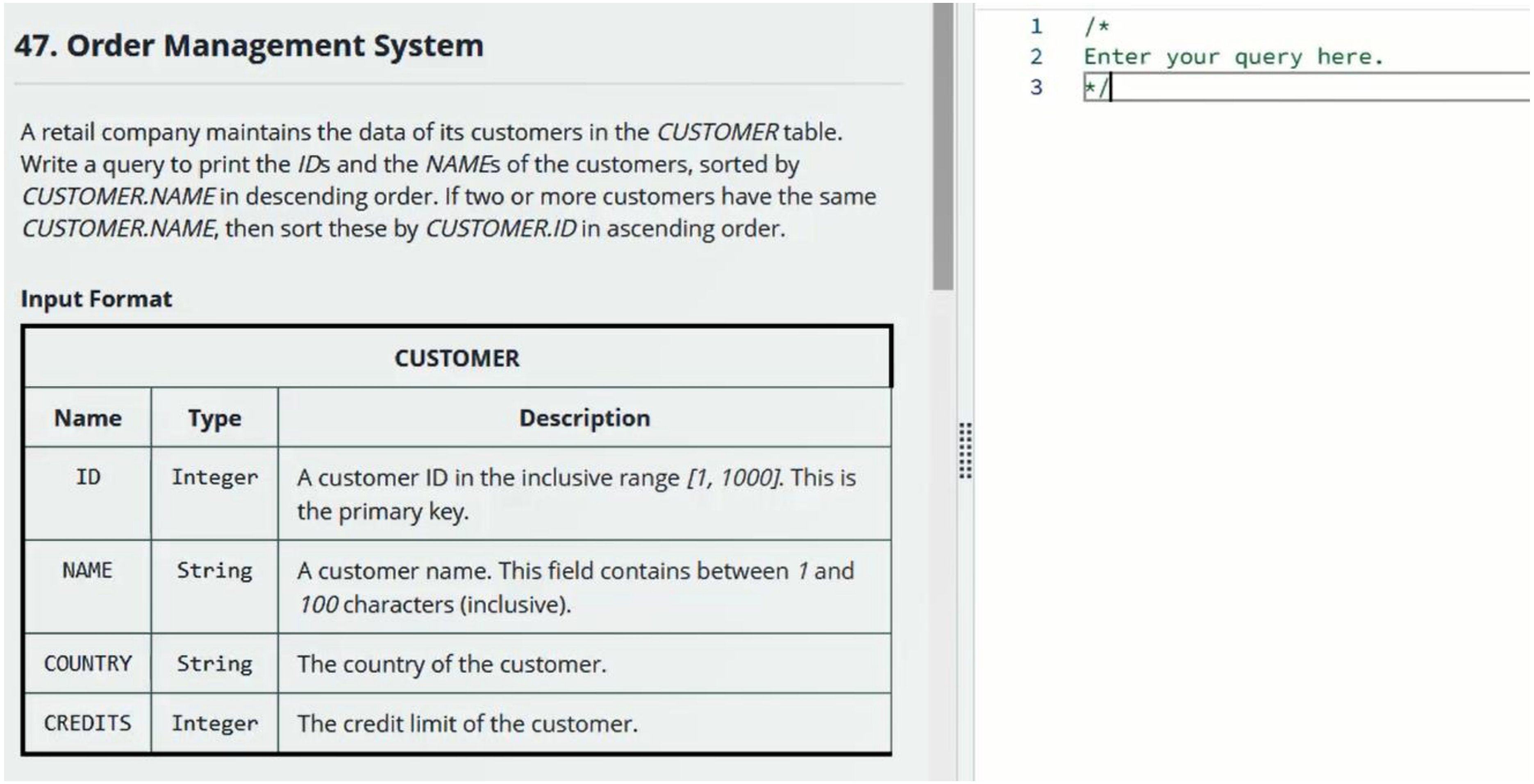
47. Order Management System

A retail company maintains the data of its customers in the *CUSTOMER* table. Write a query to print the *ID*s and the *NAME*s of the customers, sorted by *CUSTOMER.NAME* in descending order. If two or more customers have the same *CUSTOMER.NAME*, then sort these by *CUSTOMER.ID* in ascending order.

Language MySQL

● Autocomplete not supported

```
/*
Enter your query here.
*/
```



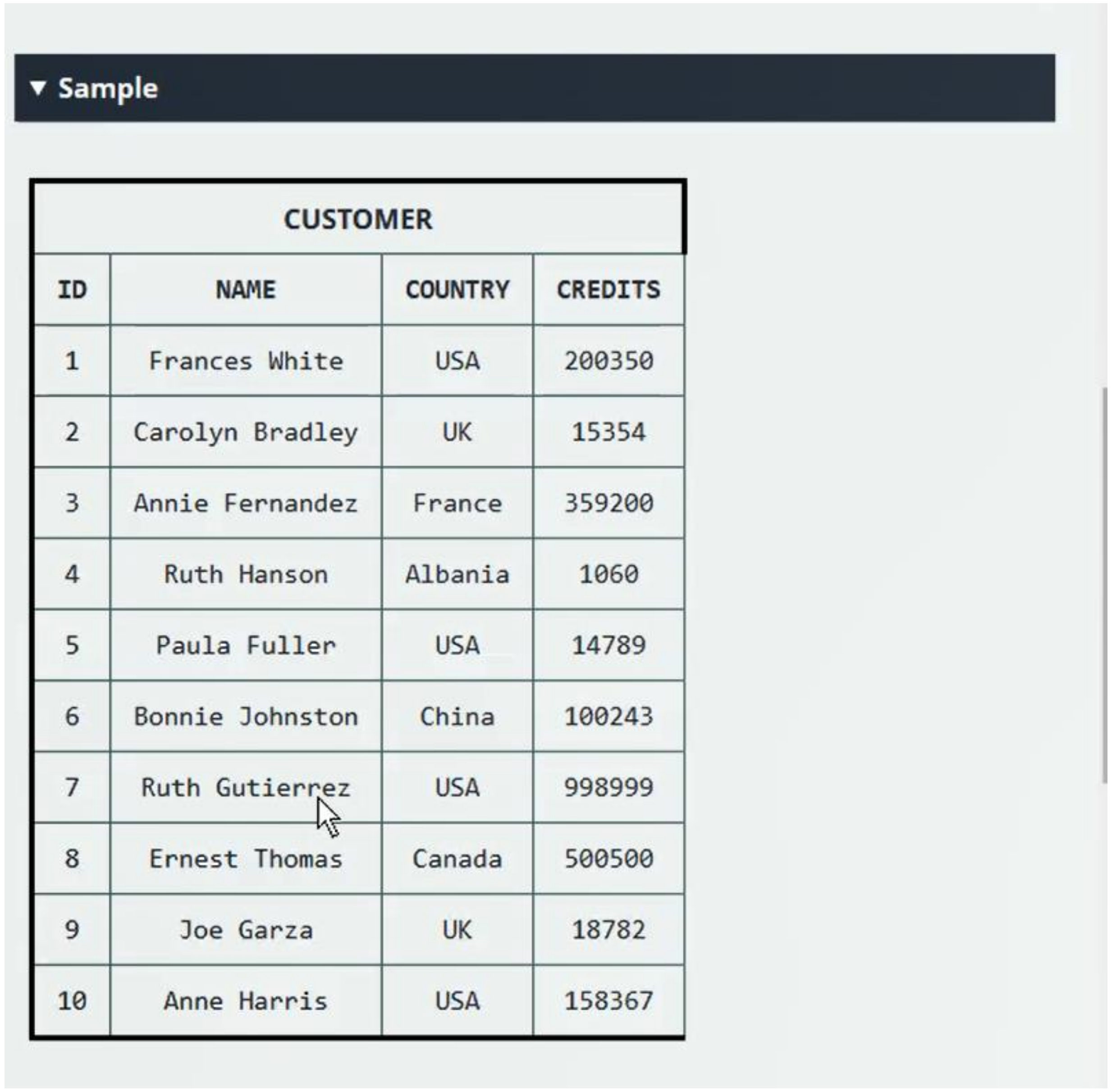
47. Order Management System

A retail company maintains the data of its customers in the *CUSTOMER* table. Write a query to print the *ID*s and the *NAME*s of the customers, sorted by *CUSTOMER.NAME* in descending order. If two or more customers have the same *CUSTOMER.NAME*, then sort these by *CUSTOMER.ID* in ascending order.

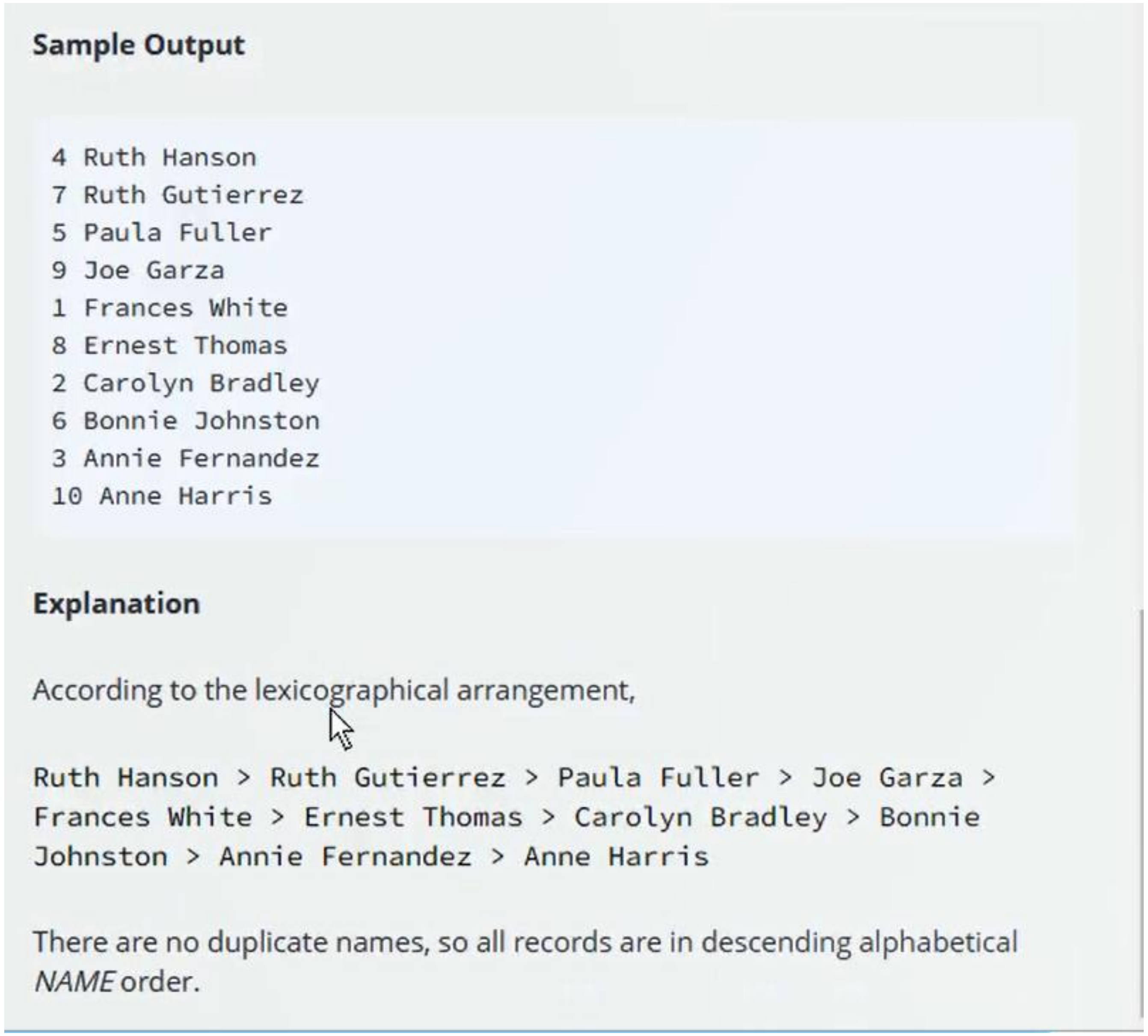
Input Format

CUSTOMER		
Name	**Type**	**Description**
ID	Integer	A customer ID in the inclusive range *[1, 1000]*. This is the primary key.
NAME	String	A customer name. This field contains between *1* and *100* characters (inclusive).
COUNTRY	String	The country of the customer.
CREDITS	Integer	The credit limit of the customer.

```
/*
Enter your query here.
*/
```


▾ Sample

CUSTOMER			
ID	NAME	COUNTRY	CREDITS
1	Frances White	USA	200350
2	Carolyn Bradley	UK	15354
3	Annie Fernandez	France	359200
4	Ruth Hanson	Albania	1060
5	Paula Fuller	USA	14789
6	Bonnie Johnston	China	100243
7	Ruth Gutierrez	USA	998999
8	Ernest Thomas	Canada	500500
9	Joe Garza	UK	18782
10	Anne Harris	USA	158367



Sample Output

```
4 Ruth Hanson
7 Ruth Gutierrez
5 Paula Fuller
9 Joe Garza
1 Frances White
8 Ernest Thomas
2 Carolyn Bradley
6 Bonnie Johnston
3 Annie Fernandez
10 Anne Harris
```

Explanation

According to the lexicographical arrangement,

```
Ruth Hanson > Ruth Gutierrez > Paula Fuller > Joe Garza >
Frances White > Ernest Thomas > Carolyn Bradley > Bonnie
Johnston > Annie Fernandez > Anne Harris
```

There are no duplicate names, so all records are in descending alphabetical *NAME* order.