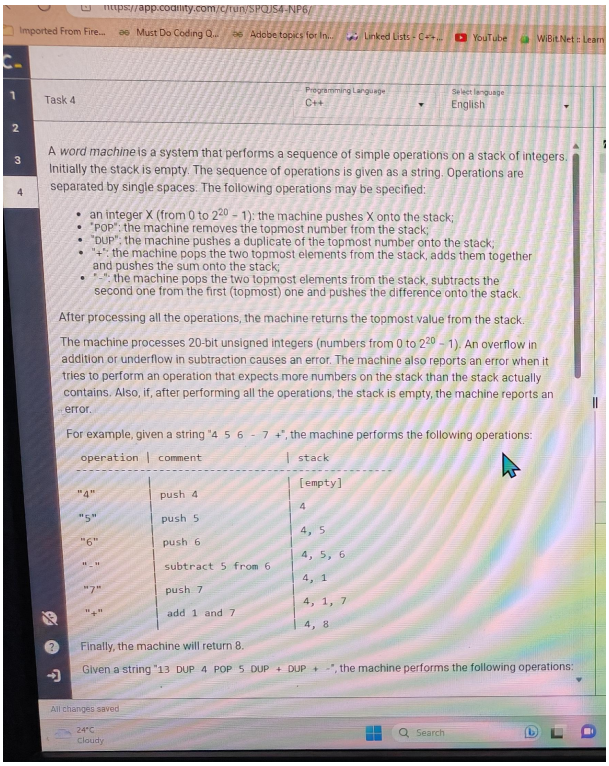


Task 4

Programming Language: C++

Select language: English

A *word machine* is a system that performs a sequence of simple operations on a stack of integers. Initially the stack is empty. The sequence of operations is given as a string. Operations are separated by single spaces. The following operations may be specified:

- an integer X (from 0 to $2^{20} - 1$): the machine pushes X onto the stack;
- "POP": the machine removes the topmost number from the stack;
- "DUP": the machine pushes a duplicate of the topmost number onto the stack;
- "+": the machine pops the two topmost elements from the stack, adds them together and pushes the sum onto the stack;
- "-": the machine pops the two topmost elements from the stack, subtracts the second one from the first (topmost) one and pushes the difference onto the stack.

After processing all the operations, the machine returns the topmost value from the stack.

The machine processes 20-bit unsigned integers (numbers from 0 to $2^{20} - 1$). An overflow in addition or underflow in subtraction causes an error. The machine also reports an error when it tries to perform an operation that expects more numbers on the stack than the stack actually contains. Also, if, after performing all the operations, the stack is empty, the machine reports an error.

For example, given a string "4 5 6 - 7 +", the machine performs the following operations:

operation	comment	stack
"4"	push 4	[empty]
"5"	push 5	4
"6"	push 6	4, 5
"-"	subtract 5 from 6	4, 5, 6
"7"	push 7	4, 1
"+"	add 1 and 7	4, 1, 7
		4, 8

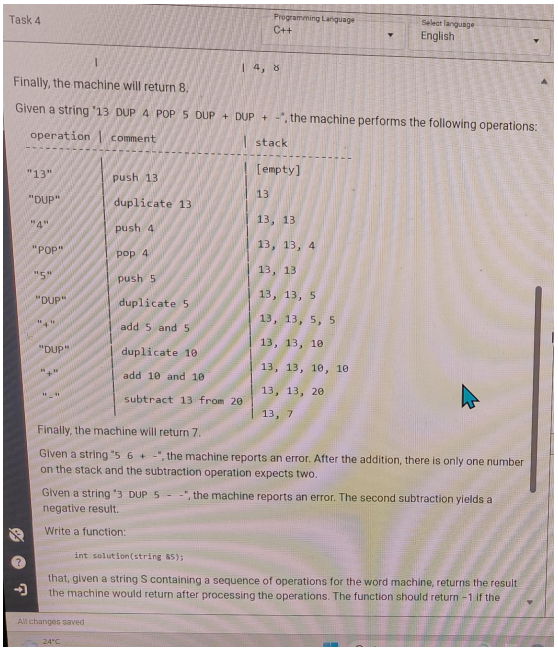
Finally, the machine will return 8.

Given a string "13 DUP 4 POP 5 DUP + DUP + -", the machine performs the following operations:

All changes saved

24°C Cloudy

Search



Task 4

Programming Language: C++

Select language: English

| 4, 8

Finally, the machine will return 8.

Given a string "13 DUP 4 POP 5 DUP + DUP + -", the machine performs the following operations:

operation	comment	stack
"13"	push 13	[empty]
"DUP"	duplicate 13	13
"4"	push 4	13, 13
"POP"	pop 4	13, 13, 4
"5"	push 5	13, 13
"DUP"	duplicate 5	13, 13, 5
"+"	add 5 and 5	13, 13, 5, 5
"DUP"	duplicate 10	13, 13, 10
"+"	add 10 and 10	13, 13, 10, 10
"-"	subtract 13 from 20	13, 13, 20
		13, 7

Finally, the machine will return 7.

Given a string "5 6 + -", the machine reports an error. After the addition, there is only one number on the stack and the subtraction operation expects two.

Given a string "3 DUP 5 - -", the machine reports an error. The second subtraction yields a negative result.

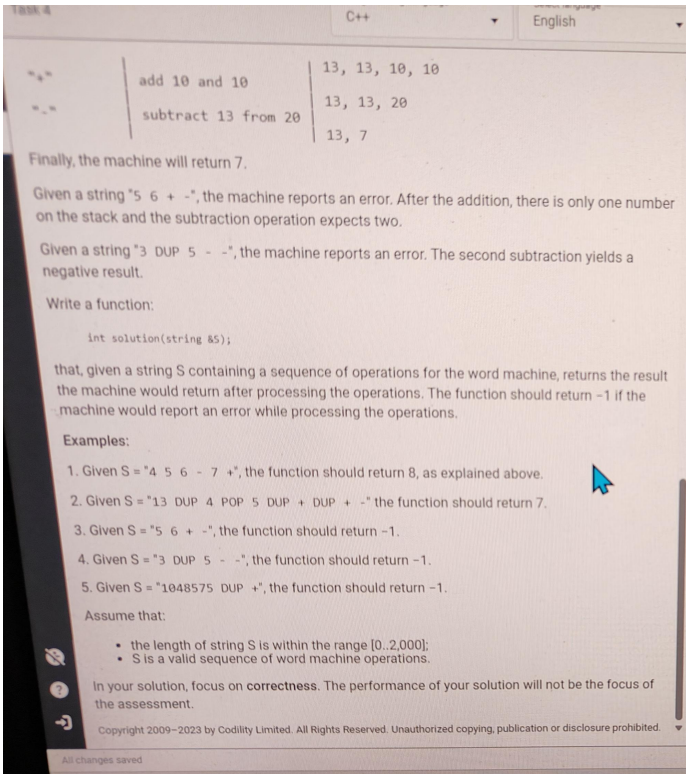
Write a function:

```
int solution(string &S);
```

that, given a string S containing a sequence of operations for the word machine, returns the result the machine would return after processing the operations. The function should return -1 if the

All changes saved

24°C



"+"	add 10 and 10	13, 13, 10, 10
"-"	subtract 13 from 20	13, 13, 20
		13, 7

Finally, the machine will return 7.

Given a string "5 6 + -", the machine reports an error. After the addition, there is only one number on the stack and the subtraction operation expects two.

Given a string "3 DUP 5 - -", the machine reports an error. The second subtraction yields a negative result.

Write a function:

```
int solution(string &S);
```

that, given a string S containing a sequence of operations for the word machine, returns the result the machine would return after processing the operations. The function should return -1 if the machine would report an error while processing the operations.

Examples:

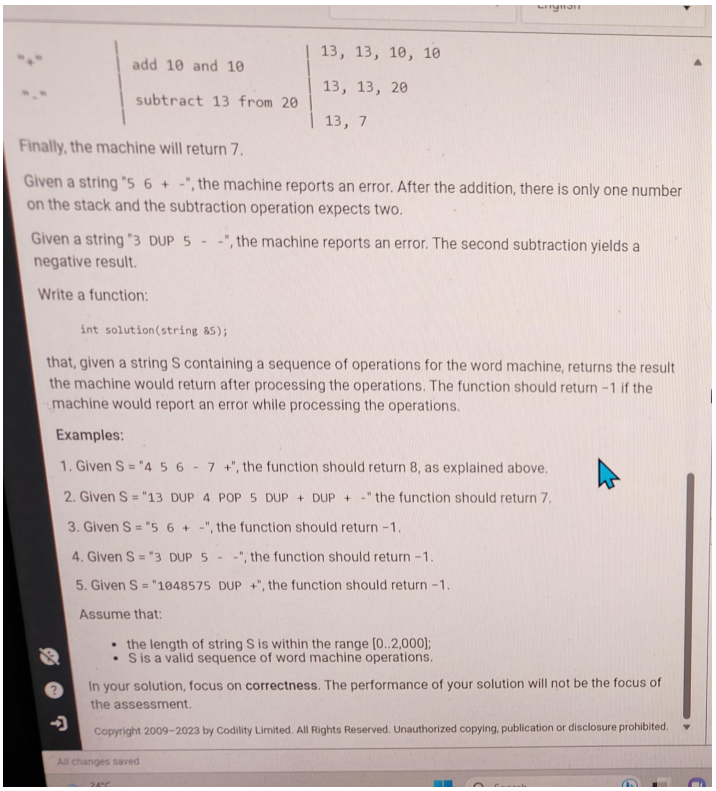
1. Given S = "4 5 6 - 7 +", the function should return 8, as explained above.
2. Given S = "13 DUP 4 POP 5 DUP + DUP + -" the function should return 7.
3. Given S = "5 6 + -", the function should return -1.
4. Given S = "3 DUP 5 - -", the function should return -1.
5. Given S = "1048575 DUP +", the function should return -1.

Assume that:

- the length of string S is within the range [0..2,000];
- S is a valid sequence of word machine operations.

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

All changes saved



"+"	add 10 and 10	13, 13, 10, 10
"-"	subtract 13 from 20	13, 13, 20
		13, 7

Finally, the machine will return 7.

Given a string "5 6 + -", the machine reports an error. After the addition, there is only one number on the stack and the subtraction operation expects two.

Given a string "3 DUP 5 - -", the machine reports an error. The second subtraction yields a negative result.

Write a function:

```
int solution(string &S);
```

that, given a string S containing a sequence of operations for the word machine, returns the result the machine would return after processing the operations. The function should return -1 if the machine would report an error while processing the operations.

Examples:

1. Given S = "4 5 6 - 7 +", the function should return 8, as explained above.
2. Given S = "13 DUP 4 POP 5 DUP + DUP + -" the function should return 7.
3. Given S = "5 6 + -", the function should return -1.
4. Given S = "3 DUP 5 - -", the function should return -1.
5. Given S = "1048575 DUP +", the function should return -1.

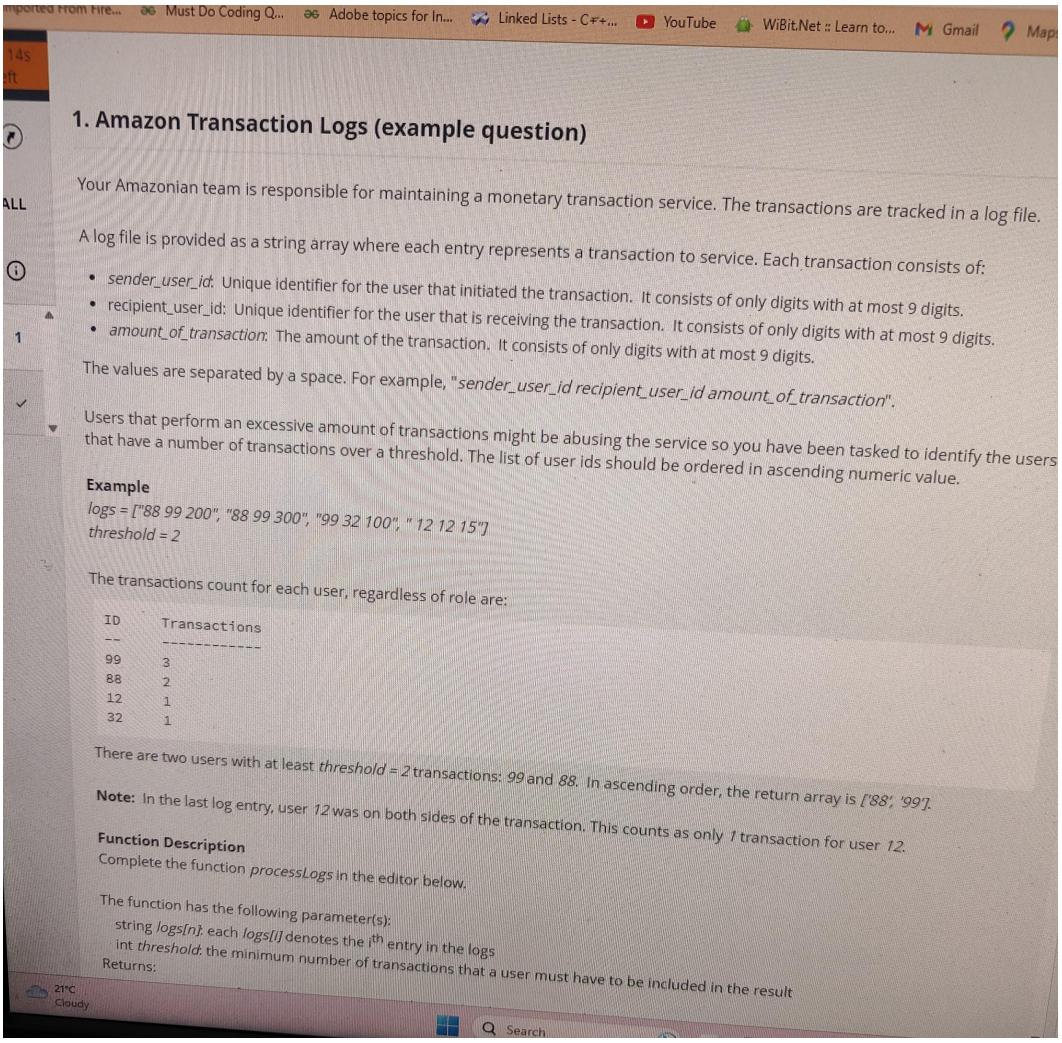
Assume that:

- the length of string S is within the range [0..2,000];
- S is a valid sequence of word machine operations.

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Copyright 2009–2023 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

All changes saved



1. Amazon Transaction Logs (example question)

Your Amazonian team is responsible for maintaining a monetary transaction service. The transactions are tracked in a log file.

A log file is provided as a string array where each entry represents a transaction to service. Each transaction consists of:

- *sender_user_id*: Unique identifier for the user that initiated the transaction. It consists of only digits with at most 9 digits.
- recipient_user_id: Unique identifier for the user that is receiving the transaction. It consists of only digits with at most 9 digits.
- *amount_of_transaction*: The amount of the transaction. It consists of only digits with at most 9 digits.

The values are separated by a space. For example, "*sender_user_id recipient_user_id amount_of_transaction*".

Users that perform an excessive amount of transactions might be abusing the service so you have been tasked to identify the users that have a number of transactions over a threshold. The list of user ids should be ordered in ascending numeric value.

Example

logs = ["88 99 200", "88 99 300", "99 32 100", " 12 12 15"]
threshold = 2

The transactions count for each user, regardless of role are:

ID	Transactions
99	3
88	2
12	1
32	1

There are two users with at least *threshold = 2* transactions: *99* and *88*. In ascending order, the return array is *['88', '99']*.

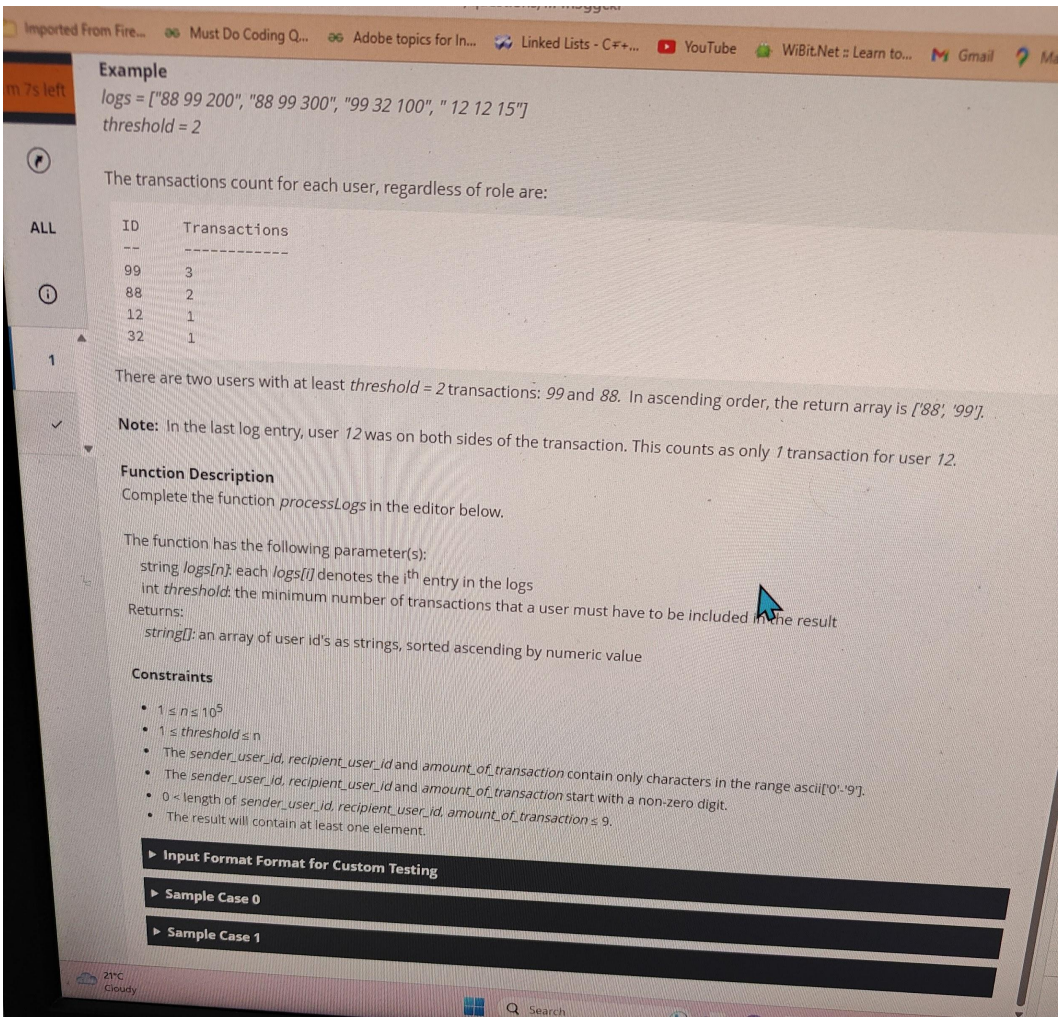
Note: In the last log entry, user *12* was on both sides of the transaction. This counts as only *1* transaction for user *12*.

Function Description

Complete the function *processLogs* in the editor below.

The function has the following parameter(s):

string *logs[n]*: each *logs[i]* denotes the i^{th} entry in the logs
int *threshold*: the minimum number of transactions that a user must have to be included in the result
Returns:



Example

logs = ["88 99 200", "88 99 300", "99 32 100", " 12 12 15"]
threshold = 2

The transactions count for each user, regardless of role are:

ID	Transactions
99	3
88	2
12	1
32	1

There are two users with at least *threshold = 2* transactions: *99* and *88*. In ascending order, the return array is *['88', '99']*.

Note: In the last log entry, user *12* was on both sides of the transaction. This counts as only *1* transaction for user *12*.

Function Description

Complete the function *processLogs* in the editor below.

The function has the following parameter(s):

string *logs[n]*: each *logs[i]* denotes the i^{th} entry in the logs
int *threshold*: the minimum number of transactions that a user must have to be included in the result
Returns:

string[]: an array of user id's as strings, sorted ascending by numeric value

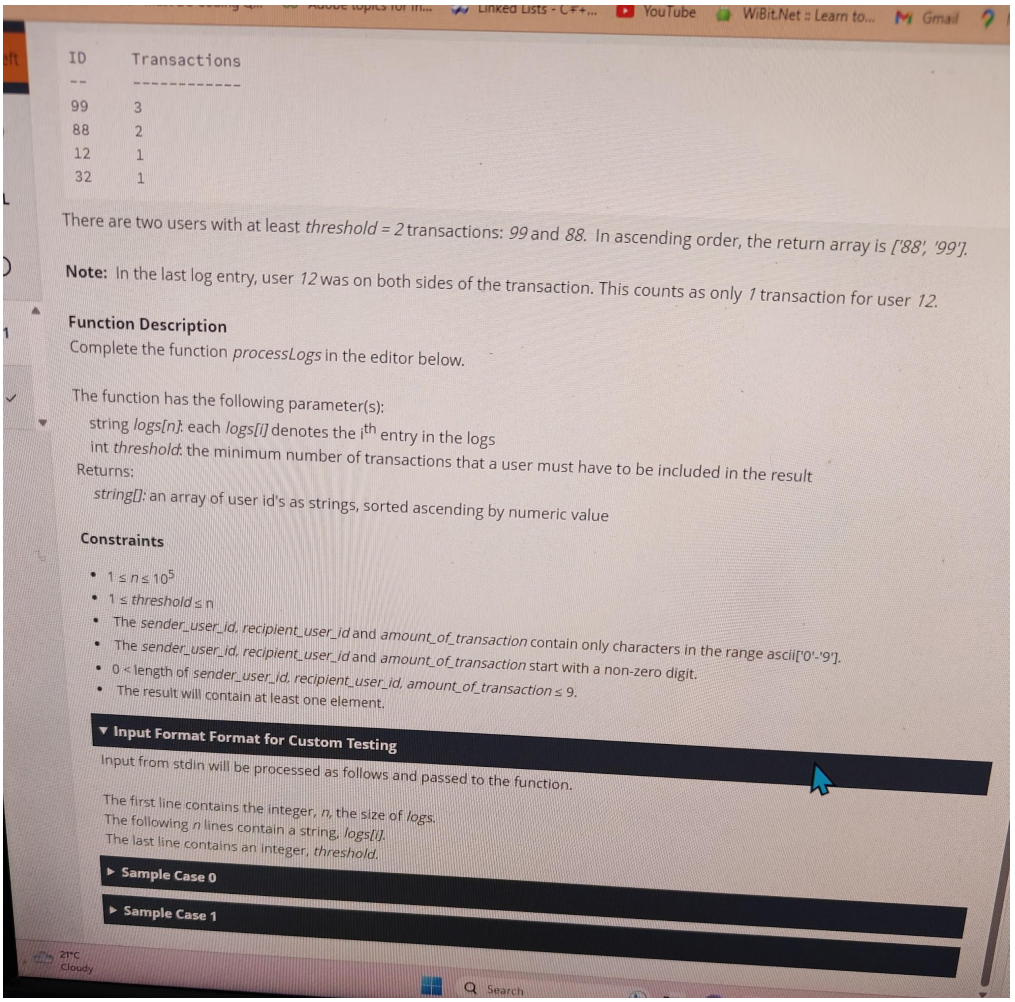
Constraints

- $1 \le n \le 10^5$
- $1 \le threshold \le n$
- The *sender_user_id*, *recipient_user_id* and *amount_of_transaction* contain only characters in the range ascii['0'-'9'].
- The *sender_user_id*, *recipient_user_id* and *amount_of_transaction* start with a non-zero digit.
- 0 < length of *sender_user_id*, *recipient_user_id*, *amount_of_transaction* ≤ 9.
- The result will contain at least one element.

▸ **Input Format Format for Custom Testing**

▸ **Sample Case 0**

▸ **Sample Case 1**



ID	Transactions
99	3
88	2
12	1
32	1

There are two users with at least *threshold = 2* transactions: *99* and *88*. In ascending order, the return array is *['88', '99']*.

Note: In the last log entry, user *12* was on both sides of the transaction. This counts as only *1* transaction for user *12*.

Function Description

Complete the function *processLogs* in the editor below.

The function has the following parameter(s):

string *logs[n]*: each *logs[i]* denotes the i^{th} entry in the logs
int *threshold*: the minimum number of transactions that a user must have to be included in the result

Returns:

string[]: an array of user id's as strings, sorted ascending by numeric value

Constraints

- $1 \le n \le 10^5$
- $1 \le threshold \le n$
- The *sender_user_id*, *recipient_user_id* and *amount_of_transaction* contain only characters in the range ascii['0'-'9'].
- The *sender_user_id*, *recipient_user_id* and *amount_of_transaction* start with a non-zero digit.
- $0 <$ length of *sender_user_id*, *recipient_user_id*, *amount_of_transaction* ≤ 9.
- The result will contain at least one element.

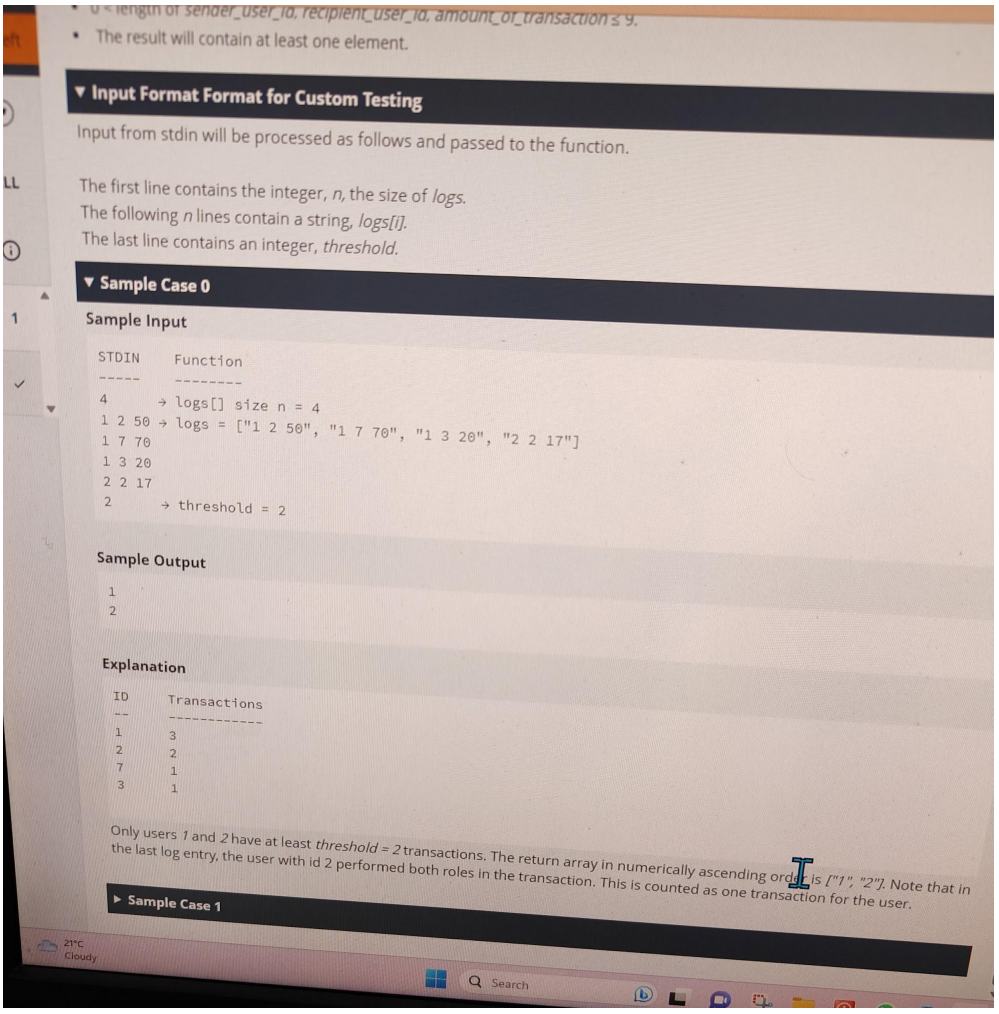
Input Format Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains the integer, *n*, the size of *logs*.
The following *n* lines contain a string, *logs[i]*.
The last line contains an integer, *threshold*.

Sample Case 0

Sample Case 1



- The result will contain at least one element.

Input Format Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains the integer, *n*, the size of *logs*.
The following *n* lines contain a string, *logs[i]*.
The last line contains an integer, *threshold*.

Sample Case 0

Sample Input

```
STDIN       Function
-----       --------
4       → logs[] size n = 4
1 2 50  → logs = ["1 2 50", "1 7 70", "1 3 20", "2 2 17"]
1 7 70
1 3 20
2 2 17
2       → threshold = 2
```

Sample Output

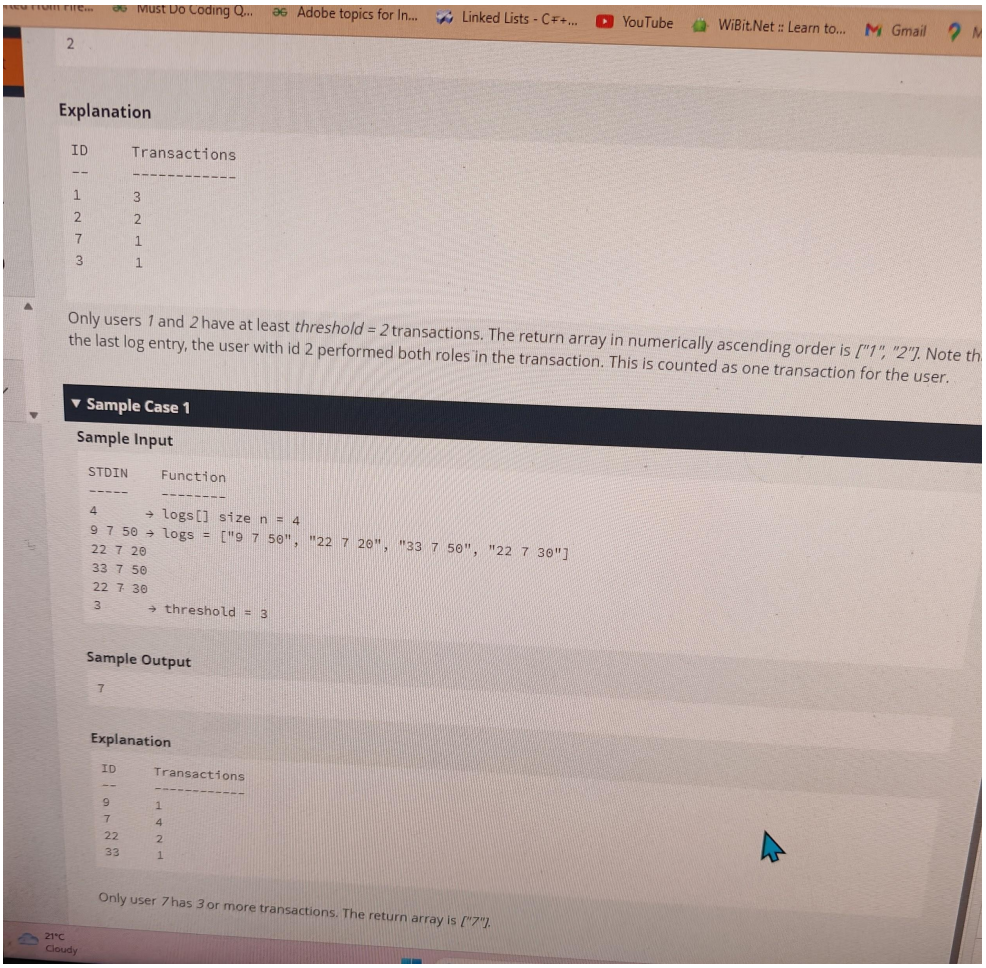
```
1
2
```

Explanation

ID	Transactions
1	3
2	2
7	1
3	1

Only users *1* and *2* have at least *threshold = 2* transactions. The return array in numerically ascending order is *["1", "2"]*. Note that in the last log entry, the user with id 2 performed both roles in the transaction. This is counted as one transaction for the user.

Sample Case 1


Must Do Coding Q...
Adobe topics for In...
Linked Lists - C++...
YouTube
WiBit.Net :: Learn to...
Gmail
2
Explanation
ID Transactions
-- ------------
1 3
2 2
7 1
3 1
Only users 1 and 2 have at least threshold = 2 transactions. The return array in numerically ascending order is ["1", "2"]. Note th
the last log entry, the user with id 2 performed both roles in the transaction. This is counted as one transaction for the user.
Sample Case 1
Sample Input
STDIN Function
----- --------
4 → logs[] size n = 4
9 7 50 → logs = ["9 7 50", "22 7 20", "33 7 50", "22 7 30"]
22 7 20
33 7 50
22 7 30
3 → threshold = 3
Sample Output
7
Explanation
ID Transactions
-- ------------
9 1
7 4
22 2
33 1
Only user 7 has 3 or more transactions. The return array is ["7"].
21°C
Cloudy