1. JavaScript: Joined Logger

In this challenge, each message object has two properties:

- property *level* having an integer value
- property *text* having a string value

For example:
```
msg = { level: 2, text: "foo" }
```

There is an implementation of a simple logger function provided that:

- takes a message object as an argument.
- writes the text of the message to the defined output.

Implement a function *joinedLogger* that:

1. takes two arguments: integer *level* and string *separator.*
2. returns a new function, *f*, such that *f* takes a variable number of message objects.
    - The function *f* uses the logger to write joined *text* values of messages that have a *level* value greater than or equal to the *level* parameter.
    - The *text* values must be joined by the *separator* string parameter, in the order they are passed to the function.

For example, let's say there are 3 defined messages:

```
msg1 = { level: 10, text: "foo" }
msg2 = { level: 20, text: "bar" }
msg3 = { level: 30, text: "baz" }
```

Calling *joinedLogger*(15, ';') must return a function *f*, such that calling *f*(msg1, msg2, msg3) causes the logger to write the string "bar;baz" to the defined output. The *level* passed to *joinedLogger* is 15, and the separator is ';'. Only ms2 and msg3 have a level greater than or equal to 15, so the text of those messages, "bar" and "baz", is joined with the ';' separator and written to the defined output by the logger.

Implementation of the function will be tested by a provided code stub on several input files. Each input file contains parameters for *joinedLogger,* followed by several values to construct messages to log. First, the *joinedLogger* function will be called with the given parameters. Then, the returned function will be called with all the messages constructed from the values given in the input. The result of the latter call will be printed to the standard output by the provided code.

▼ Input Format Format for Custom Testing

In the first line, there are two space-separated values: integer *level* and string *separator, denoting the parameters for the joinedLogger function.*
In the second line, there is an integer, n, denoting *the number of messages.*
Each of the next *n* lines contains two space-separated values: integer *level* and string *text,* denoting the level and text properties of a single message.

▼ Sample Case 0

Sample Input

```
STDIN     Function
-----     --------
21 ;    → level = 21, separator = ';'
4       → n = 4
40 foo  → msg0.level = 40, msg0.text = 'foo'
90 bar  → msg1.level = 90, msg1.text = 'bar'
20 baz  → msg2.level = 20, msg2.text = 'baz'
21 bax  → msg3.level = 21, msg3.text = 'bax'
```

Sample Output

```
foo;bar;bax
```

Explanation
The first, the second, and fourth messages have levels greater than or equal to 21. Their text values, joined with ';', are written by the logger to the defined output.

▼ Sample Case 1

Sample Input

```
STDIN     Function
-----     --------
10 -      → level = 10, separator = '-'
2         → n = 2
20 item1  → msg0.level = 20, msg0.text = 'item1'
17 item2  → msg1.level = 17, msg0.text = 'item2'
```

Sample Output

```
item1-item2
```

Explanation
Both messages have levels greater than or equal to 10. Their text values, joined with '-', are written by the logger to the defined output.