Q8. In what order will the numbers 1-4 be logged to the console when the code below is executed? Why?

```
(function() {

    console.log(1);

    setTimeout(function(){console.log(2)}, 1000);

    setTimeout(function(){console.log(3)}, 0);

    console.log(4);

})();
```

A)The numbers 1, 4, 3, and 2 will be logged to the console in the following order:

- `console.log(1);` -> This statement logs 1 to the console immediately.
- `console.log(4);` -> This statement logs 4 to the console immediately after 1.
- `setTimeout(function(){console.log(3)}, 0);` -> This sets up a timer to log 3 to the console after 0 milliseconds. However, the `setTimeout` function does not guarantee that the callback will be executed exactly after the specified delay of 0 milliseconds. Instead, it puts the callback function in the event queue, and it will be executed as soon as the call stack is empty. Since we still have more synchronous code to execute (`console.log(4)`), the event loop will wait for the call stack to be empty before executing the callback with `console.log(3)`.
- `setTimeout(function(){console.log(2)}, 1000);` -> This sets up a timer to log 2 to the console after 1000 milliseconds (1 second). However, it will not be executed

immediately, and it will be placed in the event queue like the previous `setTimeout`. The callback with `console.log(2)` will be executed after all the synchronous code and the callback with `console.log(3)` have been executed.

So, the final order of numbers logged to the console will be: 1, 4, 3, 2.