Q10. What will the following code's output be in sequence and explain why?

```
function printNumber(num) {
  console.log(num);
}

console.log(1);

setTimeout(printNumber, 0, 2);

setTimeout(printNumber, 100, 3);

console.log(4);
```

A)The output is

1
4
2
3

Explanation:

- `console.log(1);`: This statement logs `1` to the console immediately.
- `setTimeout(printNumber, 0, 2);`: This sets up a timer to call the `printNumber` function with the argument `2` after 0 milliseconds. However, even though the delay is set to 0 milliseconds, the callback is still executed after the current call stack is empty. So, it will be executed after the next line.
- `setTimeout(printNumber, 100, 3);`: This sets up a timer to call the `printNumber` function with the argument `3` after 100 milliseconds. Since there's a 100-millisecond delay specified, this callback will be executed after a short delay.

`console.log(4);`: This statement logs `4` to the console immediately after `1`.

Now, let's see the order of execution:

`console.log(1);`: Logs `1` immediately.
`console.log(4);`: Logs `4` immediately after `1`.
`setTimeout(printNumber, 0, 2);`: This timer fires after the previous synchronous code is completed. It logs `2` to the console.
`setTimeout(printNumber, 100, 3);`: This timer has a delay of 100 milliseconds. It logs `3` to the console after a 100-millisecond delay.