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);

})();
```

A8: Same as question 2, here on the start of execution of function(), 1 will be printed smoothy without any break in the normal flow of synchronous execution. But an execution context will created and pushed into the call stack corresponding to the two setTimeout function, each of them will have a timer associated to them, after the time expires, first the callback queue of console.log(3) will be placed in the callback queue and after 10 second console.log(2) will be placed in the callback queue, meanwhile console.log(4) will executed before any those two consoles. Event loop will then pop out console.log(3) into the call stack where it will get executed and after wards console.log(2) will get printed.

Hence output is

```
1
4
3
2
```