1. **Please answer the following:**
    a. What is LibUV?
    LibUV is a multi-platform support library primarily used in Node.js for asynchronous I/O operations. It provides a consistent, event-driven API for performing non-blocking input/output tasks, such as file system operations, networking, timers, and more. LibUV abstracts the complexities of working with different operating systems, making it possible for Node.js and other libraries to handle asynchronous tasks in a unified way across platforms like Linux, macOS, and Windows.

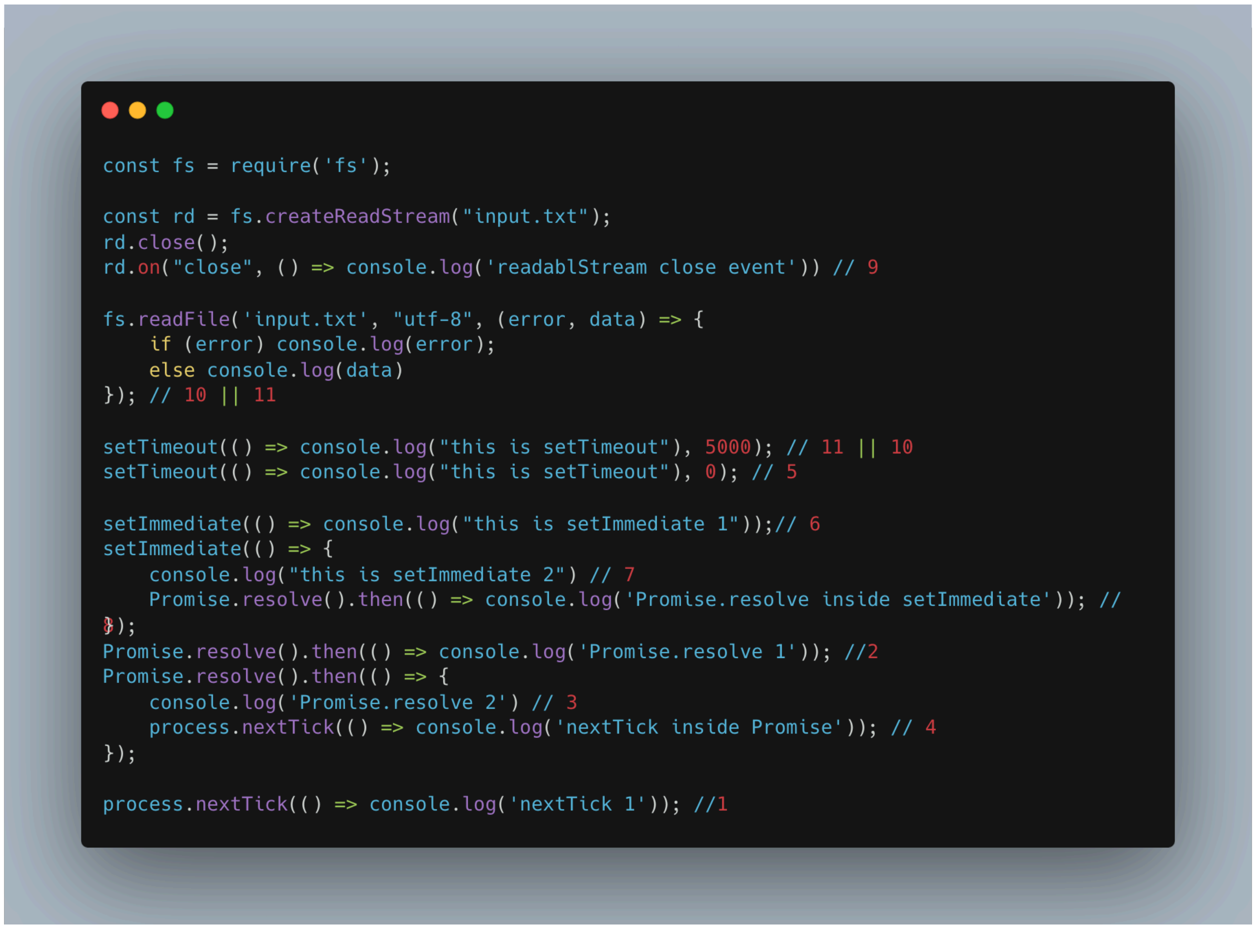
    b. Explain the difference between *setImmediate(f)* and *setTimeout(f, Time)*?

setImmediate(f)	Executes the callback *after the poll phase*, meaning as soon as the current event loop iteration finishes processing I/O callbacks, it runs. It executes as soon as possible but after the current event loop phase completes.
setTimeout(f, Time)	Schedules the callback to run in the *timers phase* in the next event loop iteration. Even though you provide 0 as the delay, it will still wait for at least one event loop iteration.

c. Explain the difference between *process.nextTick(f)* and *setImmediate(f)*?

process.nextTick(f)	This is **not** part of the event loop phases but part of the **microtask queue**. It adds the callback into the nextTick queue. Node processes all the callbacks in the nextTick queue after the current operation completes and before the event loop continues. It runs **immediately after the current phase completes** but **before the next event loop phase** begins. Even if there are timers or I/O events waiting, process.nextTick will execute first.
setImmediate(f)	This is executed in the **check phase**, which comes after t**he poll phase** where I/O callbacks are processed. It only runs after the current event loop iteration is completed, meaning it allows I/O operations and timers to run first.

2. Pls write down the output without executing the following code snippets and check it with result.



```
const fs = require('fs');

const rd = fs.createReadStream("input.txt");
rd.close();
rd.on("close", () => console.log('readablStream close event')) // 9

fs.readFile('input.txt', "utf-8", (error, data) => {
    if (error) console.log(error);
    else console.log(data)
}); // 10 || 11

setTimeout(() => console.log("this is setTimeout"), 5000); // 11 || 10
setTimeout(() => console.log("this is setTimeout"), 0); // 5

setImmediate(() => console.log("this is setImmediate 1"));// 6
setImmediate(() => {
    console.log("this is setImmediate 2") // 7
    Promise.resolve().then(() => console.log('Promise.resolve inside setImmediate')); //
8});
Promise.resolve().then(() => console.log('Promise.resolve 1')); //2
Promise.resolve().then(() => {
    console.log('Promise.resolve 2') // 3
    process.nextTick(() => console.log('nextTick inside Promise')); // 4
});

process.nextTick(() => console.log('nextTick 1')); //1
```

Step	Output	Explain
1	nextTick 1	Microtask > Process.nextTick
2	Promise.resolve 1	Microtask > Promise queue
3	Promise.resolve 2	Microtask > Promise queue
4	nextTick inside Promise	Microtask > Promise > nextTick
5	this is setTimeout	Event loop > Timer queue

6	this is setImmediate 1	Check queue
7	this is setImmediate 2	Check queue
8	Promise.resolve inside setImmediate	
9	readablStream close event	
10	Input.txt data or error message	
11	this is setTimeout	