Question 1.

LibUV is a multi-platform support library focused on providing asynchronous I/O (input/output) operations. It is primarily used to implement event loops and non-blocking I/O for networking and other asynchronous tasks. LibUV is a key component in frameworks and applications that need high-performance concurrency, such as **Node.js**.

Question 2.

In Node.js, both setImmediate(f) and setTimeout(f, time) are used to schedule functions to be executed after a certain event loop phase, but they have key differences in how and when they execute.

1. setImmediate(f)

- **When it runs**: The function passed to setImmediate(f) is executed **immediately** after the current event loop phase has completed, specifically during the **check phase** of the Node.js event loop.
- **Event Loop Timing**: The setImmediate callback is executed in the next iteration of the event loop, **right after I/O events (like timers, poll, etc.)**, but before any setTimeout or setInterval callbacks.

2. setTimeout(f, time)

- **When it runs**: The function passed to setTimeout(f, time) is executed **after the specified delay (in milliseconds)**, but **not necessarily immediately** after that delay. The delay indicates when the function should be ready to execute, but it may be delayed further depending on when the event loop is ready to handle it.
- **Event Loop Timing**: The setTimeout callback is executed during the **timers phase** of the event loop. The callback won't run until the specified time has passed, but it will only be executed when the event loop reaches the timers phase.

Question 3

The difference between **process.nextTick(f)** and **setImmediate(f)** in Node.js lies in the **timing of execution** within the **event loop**.

1. process.nextTick(f)

- **When it runs**: The function passed to process.nextTick(f) is executed **immediately** after the current operation, **before any I/O events or timers**. It runs **before the event loop continues** to the next phase.
- **Event Loop Timing**: process.nextTick(f) is executed in the **next tick** of the event loop, which happens **before** any I/O tasks or timers, even before setTimeout, setImmediate, and any other events in the loop. This gives it the highest priority.

  It is scheduled after the currently executing code completes, and **before the event loop continues**, meaning no other asynchronous operations (even setImmediate) will execute before the nextTick callback.

2. setImmediate(f)

- **When it runs**: The function passed to setImmediate(f) is executed **immediately** after the current event loop phase has completed, specifically during the **check phase** of the Node.js event loop.
- **Event Loop Timing**: The setImmediate callback is executed in the next iteration of the event loop, **right after I/O events (like timers, poll, etc.)**, but before any setTimeout or setInterval callbacks.

Question 4.

nextTick 1

Promise.resolve 1

Promise.resolve 2

nextTick inside Promise

this is setTimeout

this is setImmediate 1

this is setImmediate 2

Promise.resolve inside setImmediate

readablStream close event

this is setTimeout