Lab 12

A.

1. **LibUV** handles non-blocking I/O operations **outside** of JavaScript, in C/C++, and notifies the JS engine when they're done. LibUV is responsible for Event Loop, Thread Pool, Async I/O, and Timers

2. In **Node.js**, both setImmediate(f) and setTimeout(f, time) schedule functions to be executed asynchronously, but they differ in when the functions are executed in the **event loop**. We use setImmediate(f) when we want to execute a callback **immediately after** the current I/O event completes and setTimeout(f, time) when we want to **delay** execution to allow other operations or give the event loop a chance to process.

3. process.nextTick(f) VS setImmediate(f)

Feature	`process.nextTick()`	`setImmediate()`
Executes When	Before the event loop continues	After I/O events (check phase)
Priority	Very high	Lower than nextTick
Event Loop Phase	Microtask queue	Check phase
Can Block Loop?	Yes, if misused	No
Typical Use Case	Critical internal tasks	Deferred async callbacks

B.

Order (step-by-step)

Microtasks (synchronously scheduled):

- nextTick 1
- Promise.resolve 1
- Promise.resolve 2
- nextTick inside Promise

Poll phase (after I/O):

- readablStream close event
- <data from input.txt>

Timers phase:

- this is setTimeout (0ms delay)

Check phase:

- this is setImmediate 1
- this is setImmediate 2
- Promise.resolve inside setImmediate (microtask after setImmediate 2)

Delayed Timer:

- this is setTimeout (after 5s)

Output :

```
nextTick 1
Promise.resolve 1
Promise.resolve 2
nextTick inside Promise
readablStream close event
<contents of input.txt>     // From fs.readFile
this is setTimeout          // From setTimeout(..., 0)
this is setImmediate 1
this is setImmediate 2
Promise.resolve inside setImmediate
[this is setTimeout]
```