Q1) Illustrate the need for the WS package

The 'ws' package is a simple and efficient WebSocket implementation for Node.js. It allows for real-time, full-duplex communication between clients and servers.

Need for WS:
1. Real-time Communication – Enables instant messaging and updates without page refreshes.
2. Lightweight – Minimal overhead, making it faster and more efficient.
3. Bidirectional – Supports full-duplex communication (both client and server can send/receive data simultaneously).
4. Easy Integration – Simple API to integrate with existing Node.js servers.
5. Low Latency – Reduces response times for real-time applications.
6. Wide Use Cases – Ideal for chat apps, live feeds, gaming, and collaborative tools.

The 'ws' package is crucial for developing responsive and interactive web applications that require real-time communication.

Q2) Implement a small application which uses the WS package (Input/Output)

Below is a simple Node.js WebSocket app using the 'ws' package.

Step 1: Create server.js

```
// server.js
const WebSocket = require('ws');
const wss = new WebSocket.Server({ port: 8080 });

wss.on('connection', function connection(ws) {
  console.log('Client connected');
  ws.on('message', function message(data) {
    console.log('Received: %s', data);
    ws.send(`Hello, you said: ${data}`);
  });
});

console.log('WebSocket server is running on ws://localhost:8080');
```

Step 2: Install dependencies

```
npm init -y
npm install ws
```

Step 3: Run the server

```
node server.js
```

Client Input (Message sent via WebSocket):
"Hi server"

Output (Message from server):
"Hello, you said: Hi server"

You can test this using a WebSocket client like Postman, browser extensions, or a custom HTML file.

Q3) Illustrate the need for a code of ethics in using WS or real-time web applications

A code of ethics is essential when developing real-time applications using WebSockets to ensure secure, responsible, and respectful communication.

Importance in WS usage:
1. Data Integrity – Ensure messages are not altered or misused.
2. Security – Protect WebSocket channels from unauthorized access or data leaks.
3. Fair Use – Prevent misuse such as spamming, flooding, or broadcasting false information.
4. Transparency – Inform users of how their data is used and transmitted.
5. Respect for Users – Avoid intrusive tracking or surveillance through real-time data.

Adhering to ethical guidelines ensures trust, compliance, and user safety in real-time web communications.