node.js

1.
```
const fs = require('fs');

fs.readFile('input.txt', 'utf8', (err, data) => {
  if (err) {
    console.error('Error reading file:', err);
    return;
  }

  const modifiedContent = data.toUpperCase();

  fs.writeFile('output.txt', modifiedContent, 'utf8', (err) => {
    if (err) {
      console.error('Error writing file:', err);
      return;
    }
    console.log('File successfully written!');
  });
});
```

3.
```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const port = 3000;

app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

app.get('/', (req, res) => {
  res.send('Hello, this is a GET request!');
});


app.post('/', (req, res) => {
  const { message } = req.body;
  res.send(`Received a POST request with message: ${message}`);
});


app.listen(port, () => {
  console.log(`Server is listening on port ${port}`);
});
```

4.

```
const express = require('express');
const app = express();
const port = 3000;
const requestLoggerMiddleware = (req, res, next) => {
  console.log(‘[${new Date().toISOString()}] ${req.method} ${req.url}‘);
  next(); // Call next() to move to the next middleware or route handler
};


const errorHandlerMiddleware = (err, req, res, next) => {
  console.error(err.stack);
  res.status(500).send(’Something went wrong!’);
};

app.use(requestLoggerMiddleware);
app.use(errorHandlerMiddleware); // Register error handling middleware

app.get(’/’, (req, res) => {
  res.send(’Hello, this is a GET request!’);
});


app.listen(port, () => {
  console.log(‘Server is listening on port ${port}‘);
});
```

5.

6.

```
const express = require(’express’);
const bodyParser = require(’body-parser’);
const jwt = require(’jsonwebtoken’);

const app = express();
const port = 3000;
const secretKey = ’your_secret_key’;


const users = [
  { id: 1, username: ’john’, password: ’password123’ },
  { id: 2, username: ’jane’, password: ’password456’ }
];


app.use(bodyParser.json());


const authenticateUser = (req, res, next) => {

  const token = req.headers.authorization;
  if (!token) {
    return res.status(401).send(’Unauthorized: No token provided’);
  }
  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(401).send(’Unauthorized: Invalid token’);
    }
    req.user = decoded;
    next();
  });
};


app.post(’/login’, (req, res) => {
  const { username, password } = req.body;

  const user = users.find(user => user.username === username && user.password === password);
  if (user) {

    const token = jwt.sign({ username: user.username, id: user.id }, secretKey);
    res.json({ token });
  } else {
    res.status(401).send(’Unauthorized: Invalid username or password’);
  }
});


app.get(’/protected’, authenticateUser, (req, res) => {
  res.send(‘Welcome ${req.user.username}! This is a protected route.‘);
});

app.listen(port, () => {
  console.log(‘Server is listening on port ${port}‘);
});
```

7.

8.

```
const express = require(’express’);
const winston = require(’winston’);
const morgan = require(’morgan’);
const fs = require(’fs’);

const app = express();
const port = 3000;

// Create a log directory if it doesn’t exist
const logDir = ’./logs’;
if (!fs.existsSync(logDir)) {
  fs.mkdirSync(logDir);
}


const logger = winston.createLogger({
  level: ’info’,
  format: winston.format.combine(
    winston.format.timestamp(),
    winston.format.json()
  ),
  transports: [
    new winston.transports.Console(),
    new winston.transports.File({ filename: ‘${logDir}/error.log‘, level: ’error’ }),
    new winston.transports.File({ filename: ‘${logDir}/combined.log‘ })
  ]
});

app.use(morgan(’combined’, { stream: fs.createWriteStream(‘${logDir}/access.log‘, { flags: ’a’ }) }));


app.use((err, req, res, next) => {
  logger.error(‘${err.status || 500} - ${err.message} - ${req.originalUrl} - ${req.method} - ${req.ip}‘);
  res.status(err.status || 500).send(’Internal server error’);
});


app.get(’/error’, (req, res, next) => {
  const err = new Error(’Example error’);
  err.status = 500;
  next(err);
});


app.listen(port, () => {
  console.log(‘Server is listening on port ${port}‘);
});
```

9.

```
const fetchData1 = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {

      resolve(’Data 1’);
    }, 1000);
  });
};

const fetchData2 = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {

      resolve(’Data 2’);
    }, 1500);
  });
};


const performSequentialOperations = async () => {
  try {
    console.log(’Fetching data 1...’);
    const data1 = await fetchData1();
    console.log('Data 1:', data1);

    console.log('Fetching data 2...');
    const data2 = await fetchData2();
    console.log('Data 2:', data2);

    console.log('All operations completed successfully!');
  } catch (error) {
    console.error('An error occurred:', error);
  }
};


performSequentialOperations();
```

10.

```
const cron = require('node-cron');


cron.schedule('* * * * *', () => {

  console.log('Task executed at:', new Date());
});
```

11.