Fsd

1……>

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


Function isKaprekarNumber(number) {
   If (number === 1) return true;
   Const square = number * number;
   Const squareStr = square.toString();
   For (let I = 1; I < squareStr.length; i++) {
      Const leftPart = parseInt(squareStr.slice(0, i));
      Const rightPart = parseInt(squareStr.slice(i));
      If (leftPart + rightPart === number && leftPart !== 0 && rightPart !== 0) {
         Return true;
      }
   }
   Return false;
}


Function findKaprekarNumbers(start, end) {
   Const kaprekarNumbers = [];
   For (let I = start; I <= end; i++) {
      If (isKaprekarNumber(i)) {
         kaprekarNumbers.push(i);
      }
   }
   Return kaprekarNumbers;
}


Const start = 1;
```

```
Const end = 1000;

Const kaprekarNumbers = findKaprekarNumbers(start, end);

Fs.writeFileSync('kaprekar_numbers.txt', kaprekarNumbers.join('\n'), 'utf-8');

Console.log(`Kaprekar numbers between ${start} and ${end} have been written to
kaprekar_numbers.txt`);
```

Node kaprekar.js

2…..>

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

Const sourceFile = 'source.txt';
Const destinationFile = 'destination.txt';

// Read the content from source.txt
Fs.readFile(sourceFile, 'utf-8', (err, data) => {
  If (err) {
    Console.error(`Error reading ${sourceFile}: ${err}`);
    Return;
  }

  // Write the content to destination.txt
  Fs.writeFile(destinationFile, data, 'utf-8', (err) => {
```

```
    If (err) {
      Console.error(`Error writing to ${destinationFile}: ${err}`);
    } else {
      Console.log(`Content from ${sourceFile} copied to ${destinationFile} successfully.`);
    }
  });
});
```

Node copyfile.js

3…..?

```
Const express = require('express');
Const cookieParser = require('cookie-parser');

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

App.use(express.urlencoded({ extended: true }));
App.use(cookieParser());

// Serve HTML and CSS files from a public directory
App.use(express.static('public'));

// Render the signup form
App.get('/', (req, res) => {
  Res.sendFile(__dirname + '/public/signup.html');
```

```
});

// Handle form submission
App.post('/submit', (req, res) => {
  Const { name, contactNumber, email, address, gender, dob } = req.body;

  // Store user information in a cookie with a 15-second expiration time
  Res.cookie('registered', JSON.stringify({ name, contactNumber, email, address, gender, dob }), { maxAge: 15000 });

  // Render a confirmation message
  Res.send('Thank you for registering! <a href="/details">View Details</a>');
});

// Display user details from the cookie
App.get('/details', (req, res) => {
  Const userData = req.cookies.registered;

  If (!userData) {
    Return res.send('No user data found. <a href="/">Go Back</a>');
  }

  Const user = JSON.parse(userData);

  // Render user details and a logout link
  Res.send(`
    <h2>User Details:</h2>
    <p>Name: ${user.name}</p>
    <p>Contact Number: ${user.contactNumber}</p>
```

```
    <p>Email: ${user.email}</p>
    <p>Address: ${user.address}</p>
    <p>Gender: ${user.gender}</p>
    <p>DOB: ${user.dob}</p>
    <a href="/logout">Logout</a>
  `);
});

// Logout by clearing the registered cookie
App.get('/logout', (req, res) => {
  Res.clearCookie('registered');
  Res.redirect('/');
});

App.listen(port, () => {
  Console.log(`Server is running on port ${port}`);
});
```

Npm install express cookie-parser

Node app.js

4….>

Mkdir express-student-form

Cd express-student-form

Npm init -y

Npm install express pug

```
Const express = require('express');
Const app = express();
Const port = 3000;

// Set up Pug as the view engine
App.set('view engine', 'pug');
App.set('views', __dirname + '/views');

App.use(express.urlencoded({ extended: true }));

// Serve static files (e.g., stylesheets)
App.use(express.static('public'));

// Define a route to display the student form
App.get('/', (req, res) => {
  Res.render('student-form');
});

// Handle form submission and display submitted data
App.post('/data', (req, res) => {
  Const { rollNo, name, division, email, subject } = req.body;
  Res.render('display-data', { rollNo, name, division, email, subject });
});

App.listen(port, () => {
  Console.log(`Server is running on port ${port}`);
});
```

```
Doctype html
Html
  Head
    Title Student Form
  Body
    H1 Student Form
    Form(action=”/data”, method=”POST”)
      Label(for=”rollNo”) Roll No:
      Input(type=”number”, name=”rollNo”, required)
      Br
      Label(for=”name”) Name:
      Input(type=”text”, name=”name”, required)
      Br
      Label(for=”division”) Division:
      Input(type=”text”, name=”division”, required)
      Br
      Label(for=”email”) Email:
      Input(type=”email”, name=”email”, required)
      Br
      Label Subject:
      Input(type=”radio”, name=”subject”, value=”FSD-2”, required) FSD-2
      Input(type=”radio”, name=”subject”, value=”COA”, required) COA
      Input(type=”radio”, name=”subject”, value=”PYTHON-2”, required) PYTHON-2
      Input(type=”radio”, name=”subject”, value=”DM”, required) DM
      Input(type=”radio”, name=”subject”, value=”TOC”, required) TOC
      Br
      Input(type=”submit”, value=”Submit”)
```

```
Doctype html
Html
  Head
    Title Student Data
  Body
    H1 Student Data
    Ul
      Li Roll No: #{rollNo}
      Li Name: #{name}
      Li Division: #{division}
      Li Email: #{email}
      Li Subject: #{subject}
    A(href="/") Back to Form
```

Node app.js

5….>

```
Const express = require('express');
Const multer = require('multer');
Const path = require('path');
Const app = express();
Const port = 3000;

// Set up multer for handling file uploads
Const storage = multer.memoryStorage();
Const upload = multer({
```

```
  Storage: storage,
  Limits: { fileSize: 1024 * 1024 }, // 1MB limit
  fileFilter: (req, file, cb) => {
    // Allow only text/plain MIME type (text files)
    If (file.mimetype === 'text/plain') {
      Cb(null, true);
    } else {
      Cb(new Error('Only text files are allowed'));
    }
  },
});

App.use(express.static('public'));

// Serve the HTML form
App.get('/', (req, res) => {
  Res.sendFile(path.join(__dirname, 'public', 'upload.html'));
});

// Handle file upload
App.post('/upload', upload.single('file'), (req, res) => {
  If (!req.file) {
    Return res.status(400).send('No file uploaded.');
  }

  // Access the file contents from req.file.buffer
  Const fileContents = req.file.buffer.toString('utf-8');

  // Process the file contents here (e.g., save to a database, manipulate, etc.)
```

```
  Res.send(`File uploaded successfully. Contents: <pre>${fileContents}</pre>`);
});

App.listen(port, () => {
  Console.log(`Server is running on port ${port}`);
});
```

```
<!DOCTYPE html>
<html>
<head>
  <title>File Upload</title>
</head>
<body>
  <h1>Upload a Text File (Max 1MB)</h1>
  <form action=”/upload” method=”post” enctype=”multipart/form-data”>
    <input type=”file” name=”file” accept=”.txt” required>
    <input type=”submit” value=”Upload”>
  </form>
</body>
</html>
```

```
Npm install express multer
Node app.js
```

6…>

```
Npm init -y
Npm install express



Const express = require('express');
Const session = require('express-session');
Const app = express();
Const port = 3000;

// Set up session middleware
App.use(
  Session({
    Secret: 'mysecret', // Change this to a more secure secret key
    Resave: false,
    saveUninitialized: true,
  })
);

// Serve static files (HTML, CSS, etc.)
App.use(express.static('public'));

// Handle form submission and save the username in session
App.post('/savesession', (req, res) => {
  Const { username } = req.body;
  Req.session.username = username;
  Res.redirect('/fetchsession');
});

// Display session value and logout link
```

```
App.get('/fetchsession', (req, res) => {
  Const username = req.session.username;
  If (!username) {
    Res.redirect('/');
  } else {
    Res.send(`
      <h1>Welcome, ${username}!</h1>
      <a href="/deletesession">Logout</a>
    `);
  }
});

// Handle session deletion and redirect to the index page
App.get('/deletesession', (req, res) => {
  Req.session.destroy((err) => {
    If (err) {
      Console.error(err);
    }
    Res.redirect('/');
  });
});

App.listen(port, () => {
  Console.log(`Server is running on port ${port}`);
});
```

Node app.js

7….>

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple HTML Page</title>
</head>
<body>
  <h1>Hello, World!</h1>
  <p>This is a simple HTML page.</p>
</body>
</html>
```

```
Const http = require('http');
Const fs = require('fs');
Const path = require('path');

Const server = http.createServer((req, res) => {
  // Define the path to the HTML file
  Const filePath = path.join(__dirname, 'simple.html');

  // Check if the request URL is for the HTML file
  If (req.url === '/simple.html') {
    // Read the HTML file
    Fs.readFile(filePath, 'utf-8', (err, data) => {
      If (err) {
        Res.writeHead(500, { 'Content-Type': 'text/plain' });
        Res.end('Internal Server Error');
        Return;
```

```
        }

        // Set the response headers and send the HTML content
        Res.writeHead(200, { 'Content-Type': 'text/html' });
        Res.end(data);
      });
  } else {
    // Handle other requests (e.g., 404 Not Found)
    Res.writeHead(404, { 'Content-Type': 'text/plain' });
    Res.end('Not Found');
  }
});

Const port = 3000;
Server.listen(port, () => {
  Console.log(`Server is running on http://localhost:${port}`);
});
```

Node server.js

8…..>

Pug file

```
Doctype html
Html
 Head
  Title Online Store
  Link(rel='stylesheet', href='/styles.css')
 Body
```

```
  H1 Welcome to Our Online Store
  H2 Products
  Ul
    Each product in products
      Li
        A(href=`/products/${product.id}`)= product.name
  A(href='/') Go Back Home
```

Pug file

```
Doctype html
Html
  Head
    Title Error
    Link(rel='stylesheet', href='/styles.css')
  Body
    H1 Error 404 – Page Not Found
    P The page you are looking for does not exist.
    A(href='/') Go Back Home
```

```
Const express = require('express');
Const app = express();
Const port = 3000;

// Define an array of product objects
Const products = [
  { id: 1, name: 'Product 1', description: 'Description 1', price: '$10' },
  { id: 2, name: 'Product 2', description: 'Description 2', price: '$20' },
  { id: 3, name: 'Product 3', description: 'Description 3', price: '$30' },
];
```

```
// Set the view engine and views directory
App.set('view engine', 'pug');
App.set('views', __dirname + '/views');

// Serve static files from the public directory
App.use(express.static('public'));

// Define a route to display a welcome message on the homepage
App.get('/', (req, res) => {
  Res.send('Welcome to our online store!');
});

// Define a route to display a list of products
App.get('/products', (req, res) => {
  Res.render('products', { products });
});

// Define a dynamic route for product details
App.get('/products/:id', (req, res) => {
  Const productId = parseInt(req.params.id);
  Const product = products.find((p) => p.id === productId);

  If (!product) {
    Res.status(404).render('error');
    Return;
  }

  Res.send(`Product Details: ${product.name}, ${product.description}, Price: ${product.price}`);
```

```
});

// Handle 404 errors
App.use((req, res) => {
  Res.status(404).render('error');
});

App.listen(port, () => {
  Console.log(`Server is running on port ${port}`);
});
```

Node app.js

11….>

```
Let variable1 = 0;
Let variable2 = 0;

Function incrementAndDisplay() {
  Variable1++;
  Variable2++;
  Const sum = variable1 + variable2;
  Console.log(`Variable 1: ${variable1}, Variable 2: ${variable2}, Sum: ${sum}`);
}

// Call the incrementAndDisplay function every 1 second (1000 milliseconds)
setInterval(incrementAndDisplay, 1000);
```

12…>

Mkdir express-student-form

Cd express-student-form

Npm init -y

Npm install express pug

Pug file

```
Doctype html
Html
  Head
    Title Student Form
  Body
    H1 Student Form
    Form(action=”/student”, method=”POST”)
      Label(for=”name”) Name:
      Input(type=”text”, name=”name”, required)
      Br
      Label(for=”email”) Email:
      Input(type=”email”, name=”email”, required)
      Br
      Label Course:
      Input(type=”radio”, name=”course”, value=”CE”, required) CE
      Input(type=”radio”, name=”course”, value=”IT”, required) IT
      Input(type=”radio”, name=”course”, value=”CSE”, required) CSE
      Br
      Input(type=”submit”, value=”Submit”)
```

Create a Pug file to display the submitted data (views/student.pug):

Pug file

```
Doctype html
Html
  Head
    Title Student Data
  Body
    H1 Student Data
    P Name: #{name}
    P Email: #{email}
    P Course: #{course}
    A(href="/") Back to Form
```

Create an Express.js application in app.js:

Javascript:

```
Const express = require('express');
Const app = express();
Const port = 3000;

App.set('view engine', 'pug');
App.set('views', __dirname + '/views');

App.use(express.urlencoded({ extended: true }));

App.use(express.static('public'));

App.get('/', (req, res) => {
  Res.render('student-form');
});
```

```
App.post('/student', (req, res) => {
  Const { name, email, course } = req.body;
  Res.render('student', { name, email, course });
});

App.listen(port, () => {
  Console.log(`Server is running on port ${port}`);
});
```

Node app.js

13….>

```
Const readline = require('readline');

Const rl = readline.createInterface({
  Input: process.stdin,
  Output: process.stdout
});

// Function to calculate the area of a circle
Function calculateCircleArea(radius) {
  Return Math.PI * Math.pow(radius, 2);
}

// Function to calculate the perimeter of a square
Function calculateSquarePerimeter(side) {
  Return 4 * side;
}
```

```
// Prompt user for the radius of the circle
Rl.question('Enter the radius of the circle: ', (radiusInput) => {
  Const radius = parseFloat(radiusInput);

  If (isNaN(radius) || radius < 0) {
    Console.log('Radius must be positive.');
  } else {
    Const circleArea = calculateCircleArea(radius);
    Console.log(`The area of the circle is: ${circleArea.toFixed(2)}`);
  }

  // Prompt user for the side of the square
  Rl.question('Enter the side of the square: ', (sideInput) => {
    Const side = parseFloat(sideInput);

    If (isNaN(side) || side < 0) {
      Console.log('Side must be positive.');
    } else {
      Const squarePerimeter = calculateSquarePerimeter(side);
      Console.log(`The perimeter of the square is: ${squarePerimeter}`);
    }

    // Close the readline interface
    Rl.close();
  });
});
```

Node geometry.js

14…>

```
Const EventEmitter = require('events');

// Create a custom event emitter
Const myEmitter = new EventEmitter();

// Listener 1
Function listener1() {
  Console.log('Listener 1 called');
}

// Listener 2
Function listener2() {
  Console.log('Listener 2 called');
}

// Add the listeners to the common event
myEmitter.on('commonEvent', listener1);
myEmitter.on('commonEvent', listener2);

// Print the number of listeners associated with the emitter
Console.log(`Number of listeners: ${myEmitter.listenerCount('commonEvent')}`);

// Emit the common event, which will trigger both listeners
myEmitter.emit('commonEvent');

// Remove one of the listeners (listener2)
```

```
myEmitter.removeListener('commonEvent', listener2);

// Print the number of remaining listeners
Console.log(`Number of remaining listeners: ${myEmitter.listenerCount('commonEvent')}`);

// Emit the common event again, which will trigger only listener1
myEmitter.emit('commonEvent');
```

15…>

```
Const express = require('express');
Const multer = require('multer');
Const app = express();

// Define storage for uploaded files
Const storage = multer.diskStorage({
  Destination: 'uploads/', // Specify the destination directory
  Filename: (req, file, cb) => {
    // Customize the filename as needed (e.g., keep the original filename)
    Const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);
    Cb(null, file.fieldname + '-' + uniqueSuffix + '.' + file.originalname.split('.').pop());
  },
});

// Initialize multer with the defined storage configuration
Const upload = multer({ storage });

// Create a route to handle file uploads
App.post('/upload', upload.single('file'), (req, res) => {
```

```
  // Handle the uploaded file here (e.g., save it to a database, perform further processing)
  Res.send('File uploaded successfully');
});

// Start the Express.js server
Const port = 3000;
App.listen(port, () => {
  Console.log(`Server is running on port ${port}`);
})
```

9….>

```
Mkdir weather-forecast-app
Cd weather-forecast-app
Npm init -y
Npm install express pug
```

Pug file

```
Doctype html
Html
  Head
    Title Weather Forecast
  Body
    H1 Welcome to the Weather Forecast Service
    P Please enter a location to check the weather.
    Form(action="/weather", method="get")
      Input(type="text", name="location", placeholder="Enter location", required)
      Button(type="submit") Get Weather
```

Pug file

```
Doctype html
Html
  Head
    Title Weather Forecast
  Body
    H1 Weather Forecast for #{location}
    P Temperature: #{temperature}Â°C
    P Description: #{description}
    A(href="/") Go Back
```

Avascript

Copy code

```
Const express = require('express');
Const app = express();
Const port = 3000;

// Set the view engine and views directory
App.set('view engine', 'pug');
App.set('views', __dirname + '/views');

// Serve static files from the public directory (e.g., CSS, images)
App.use(express.static('public'));

// Define a route for the root URL ("/") to display the welcome message
App.get('/', (req, res) => {
  Res.render('index');
});
```