;

1) **Registration Form Class Component**

```
// RegistrationForm.js
import React, { Component } from 'react';
class RegistrationForm extends Component {
render() {
return (
<div>
<h2>Register</h2>
<form>
<input name="name" placeholder="Name" /><br />
<input name="email" type="email" placeholder="Email" /><br />
<input name="password" type="password" placeholder="Password" /><br />
<button type="submit">Submit</button>
</form>
</div>
);}}
export default RegistrationForm;
```

2) **Program for creating registration form using react.js function component**

```
// RegistrationForm.js
import React from 'react';
function RegistrationForm() {
  return (
    <div>
      <h2>Register</h2>
      <form>
        <input name="name" placeholder="Name" /><br />
        <input name="email" type="email" placeholder="Email" /><br />
        <input name="password" type="password" placeholder="Password" /><br />
        <button type="submit">Submit</button>
      </form>
    </div>
  );
}
export default RegistrationForm;
```

3) program for applying CSS style in react.js application

App.css

```
.app-container
{ text-align: center; margin: 20px; }

.title { color: blue; font-size: 24px; }

.description { color: gray; font-size: 16px; }
```

App.js

```
// App.js
import React from 'react';
import './App.css';

function App() {
  return (
    <div className="app-container">
      <h2 className="title">Styled React Application</h2>
      <p className="description">This is a simple React application with applied CSS styles.</p>
    </div>
  );
}

export default App;
```

4) program for display any 5 MUI Components

```
import React from 'react';

import { Button, TextField, Checkbox, Switch, Typography } from '@mui/material';



const App = () => (

  <div>

    <Typography variant="h4">MUI Components</Typography>

    <Button variant="contained">Click Me</Button>

    <TextField label="Input" variant="outlined" />

    <Checkbox />

    <Switch />

  </div>

);


export default App;
```

5) program for printing hello on the browser using Node.js web module.

```
const http = require('http');

http.createServer((req, res) => {

   // Set the HTTP status code and content type

   res.writeHead(200, { 'Content-Type': 'text/plain' });

   // Send the response body

   res.end('Hello World');

}).listen(3000, () => {

   console.log('Server running on http://localhost:3000');

});
```

6) program for printing hello on the browser using Node.js web module.

```
function greet(name, callback)
 {
  console.log(`Hello, ${name}`);
  callback();
}

greet('Alice', () => console.log('Callback executed!'));
```

7) program to read the file contents using Node.js file system

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


fs.readFile('example.txt', 'utf8', (err, data) => {

  if (err) throw err;

  console.log(data);

});
```

8) program to write the contents to the file using Node.js file system

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


fs.writeFile('example.txt', 'Hello World!', (err) => {

  if (err) throw err;

  console.log('File written successfully!');

});
```

9) **program to read the contents from the directory and display on console using Node.js**

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


fs.readdir('.', (err, files) => {
  if (err) throw err;
  console.log(files);
});
```

10) **program for demonstrating any 5 functions of file systems**

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


fs.rename('old.txt', 'new.txt', console.log);
fs.unlink('new.txt', console.log);
fs.mkdir('test', console.log);
fs.rmdir('test', console.log);
fs.stat('example.txt', console.log);
```

11) **program for demonstrating any 5 functions of console global object**

```
console.log("This is a log message");
console.error("This is an error message");
console.warn("This is a warning");
console.time("Timer");
console.timeEnd("Timer");
console.assert(5 > 10, "Assertion failed!");
```

12) program for demonstrating any 5 functions of process global object

```
console.log("This is a log message");

console.error("This is an error message");

console.warn("This is a warning");

console.time("Timer");

console.timeEnd("Timer");

console.assert(5 > 10, "Assertion failed!");
```

13) program for demonstrating any 5 functions of OS utility module

```
const os = require('os');

// 1. os.platform() const os = require('os');

console.log('Operating system platform:', os.platform());

// 2. os.arch()

console.log('CPU architecture:', os.arch());

// 3. os.cpus()

console.log('CPU Info:', os.cpus());

// 4. os.freemem()

console.log('Free memory:', os.freemem(), 'bytes');

// 5. os.totalmem()

console.log('Total memory:', os.totalmem(), 'bytes');
```

14) program for demonstrating any 5 functions of Path utility module

```
const path = require('path');

// 1. path.join()

console.log('Joined path:', path.join('folder1', 'folder2', 'file.txt'));

// 2. path.resolve()

console.log('Resolved path:', path.resolve('folder1', 'folder2', 'file.txt'));

// 3. path.basename()

console.log('Base name:', path.basename('/folder1/folder2/file.txt'));

// 4. path.extname()

console.log('File extension:', path.extname('/folder1/folder2/file.txt'));

// 5. path.dirname()

console.log('Directory name:', path.dirname('/folder1/folder2/file.txt'));
```

15) program for demonstrating any 5 functions of Net utility module

```
const net = require('net');
// 1. net.createServer()
const server = net.createServer((socket) => {
  console.log('Client connected');
  socket.write('Hello from server');
  socket.end();
});
server.listen(8080, () => {
  console.log('Server is listening on port 8080');
});
// 2. socket.write()
const client = net.createConnection({ port: 8080 }, () => {
  console.log('Connected to server');
  client.write('Hello from client');
});
// 3. socket.end()
client.on('data', (data) => {
  console.log('Received from server:', data.toString());
  client.end(); // Close the connection after receiving data
});
// 4. server.listen()
// Already demonstrated in the `net.createServer()` part
// 5. net.isIP()
console.log('Is "127.0.0.1" an IP address?', net.isIP('127.0.0.1')); // Should return 4 (IPv4)
console.log('Is "localhost" an IP address?', net.isIP('localhost')); // Should return 0 (not an IP)
```

16) program for demonstrating any 3 functions of DNS utility module

```
const dns = require('dns');

dns.lookup('example.com', (err, address) => console.log('IP Address:', address));
dns.resolve4('example.com', (err, addresses) => console.log('IPv4 Addresses:', addresses));
dns.reverse('93.184.216.34', (err, hostnames) => console.log('Hostnames:', hostnames));
```

17) program for reading data from stream using Node.js

```
const fs = require('fs');
const readStream = fs.createReadStream('example.txt', 'utf8');
readStream.on('data', (chunk) => console.log('Data:', chunk));
```

18) program for writing data to the stream using Node.js

```
const fs = require('fs');
const writeStream = fs.createWriteStream('output.txt');
writeStream.write('Hello, Stream!');
writeStream.end();
```

19) program for creating a module for arithmetic operations and use it in another program using Node.js

```
//arithmetic.js
exports.add = (a, b) => a + b;
exports.subtract = (a, b) => a - b;
exports.multiply = (a, b) => a * b;
exports.divide = (a, b) => b !== 0 ? a / b : 'Error: Division by zero';

// app.js
const arithmetic = require('./arithmetic');
console.log('Add:', arithmetic.add(5, 3));
console.log('Subtract:', arithmetic.subtract(5, 3));
console.log('Multiply:', arithmetic.multiply(5, 3));
console.log('Divide:', arithmetic.divide(5, 3));
```

20) program for demonstrating any 5 functions of request object in Express.js

```
const express = require('express');
const app = express();
app.get('/', (req, res) => {
  console.log("URL:", req.url);
  console.log("Method:", req.method);
  console.log("Headers:", req.headers);
  console.log("Query:", req.query);
  console.log("IP:", req.ip);
  res.send('Request Object Functions');
});
app.listen(3000, () => console.log('Server running on http://localhost:3000));
```

21) program for demonstrating any 5 functions of response object in Express.js

```
const express = require('express');
const app = express();
app.get('/json', (req, res) => {
  res.status(200).json({ message: "Hello JSON" });
});
app.get('/', (req, res) => {
  res.status(200);
  res.set('Content-Type', 'text/html');
  res.cookie('name', 'value');
  res.send('<h1>Response Object</h1>');
});
app.listen(3000, () => console.log('Server running on http://localhost:3000'));
```

22) program for get HTTP method using Express.js

```
const express = require('express');
const app = express();
app.get('/', (req, res) => {
  res.send('Welcome to the root route');
});
app.get('/get', (req, res) => {
  res.send('GET Request');
});
app.listen(3000, () => console.log('Server running on http://localhost:3000'));
```

23) program for post HTTP method using Express.js

```
const express = require('express');
const app = express();
app.use(express.json());
app.post('/post', (req, res) => {
  res.send(`POST Data: ${JSON.stringify(req.body)}`);
});
app.get('/', (req, res) => {
  res.send('Hello, this is the root route!');
});
app.listen(3000, () => console.log('Server running on http://localhost:3000'));
```

24) program for put HTTP method using Express.js

```
const express = require('express');
const app = express();
app.use(express.json());
app.get('/', (req, res) => {
  res.send('Welcome to the server!');
});
app.put('/put', (req, res) => {
  res.send(`PUT Data: ${JSON.stringify(req.body)}`);
});
app.listen(3000, () => console.log('Server running on http://localhost:3000'));
```

25) program for demonstrating the use of app.use() in Express.js

```
const express = require('express');
const app = express();
app.use((req, res, next) => {
  console.log('Middleware executed');
  next();
});
app.get('/', (req, res) => res.send('Hello World'));
app.listen(3000, () => console.log('Server running on http://localhost:3000/'));
```

27) Create the MongoDB database and insert the records either using interface or using Node.js program

```
use myDatabase
db.myCollection.insertOne({ name: 'John', age: 30 });
```

Using Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = new MongoClient('mongodb://localhost:27017');
  await client.connect();
  const db = client.db('myDatabase');
  const result = await db.collection('myCollection').insertOne({ name: 'John', age: 30 });
  console.log(result);
  client.close();
}
main();
```