Node Js Assignments

1.

```
const fs = require('fs')
const data = fs.existsSync('notes.json') ?
JSON.parse(fs.readFileSync('notes.json')) : []
const cmd = process.argv[2]

if (cmd === 'add') {
  const title = process.argv[4]
  const body = process.argv[6]
  if (!data.find(n => n.title === title)) {
    data.push({ title, body })
    fs.writeFileSync('notes.json', JSON.stringify(data))
    console.log('Note added!')
  } else {
    console.log('Duplicate title!')
  }
} else if (cmd === 'list') {
  data.forEach(n => console.log(n.title))
} else if (cmd === 'remove') {
  const title = process.argv[4]
  const filtered = data.filter(n => n.title !== title)
  fs.writeFileSync('notes.json', JSON.stringify(filtered))
  console.log('Note removed!')
}
```

2.

App.js

```
const notes = require('./notes')
const cmd = process.argv[2]

if (cmd === 'add') notes.addNote(process.argv[4], process.argv[6])
if (cmd === 'list') notes.listNotes()
if (cmd === 'remove') notes.removeNote(process.argv[4])
```

notes.js

```
const fs = require('fs')
const getData = () => fs.existsSync('notes.json') ? JSON.parse(fs.readFileSync('notes.json')) : []

function addNote(title, body) {
  const data = getData()
  if (!data.find(n => n.title === title)) {
    data.push({ title, body })
    fs.writeFileSync('notes.json', JSON.stringify(data))
    console.log('Note added!')
  }
}

function listNotes() {
  getData().forEach(n => console.log(n.title))
}
```

```
function removeNote(title) {
  const data = getData().filter(n => n.title !== title)
  fs.writeFileSync('notes.json', JSON.stringify(data))
  console.log('Note removed!')
}

module.exports = { addNote, listNotes, removeNote }
```

3.

```
import chalk from 'chalk'; // If using ES module syntax

console.log(chalk.green('Note added!'));
console.log(chalk.red('Duplicate note!'));
console.log(chalk.blue('Reading note...'));
console.log(chalk.yellow('Removing note...'));
```

4.

```
// app.js
import yargs from 'yargs';
import { hideBin } from 'yargs/helpers';

const argv = yargs(hideBin(process.argv))
```

```
.command('add', 'Add a new note', {
  title: {
    describe: 'Note title',
    demandOption: true,
    type: 'string',
  },
  body: {
    describe: 'Note body',
    demandOption: true,
    type: 'string',
  },
})
.command('remove', 'Remove a note', {
  title: {
    describe: 'Note title to remove',
    demandOption: true,
    type: 'string',
  },
})
.command('list', 'List all notes')
.command('read', 'Read a note', {
  title: {
    describe: 'Note title to read',
    demandOption: true,
    type: 'string',
  },
})
.help()
.argv;
```

```
if (argv._[0] === 'add') {
  console.log(chalk.green(`Adding note: ${argv.title}`));
} else if (argv._[0] === 'remove') {
  console.log(chalk.yellow(`Removing note: ${argv.title}`));
}
```

5.

```
const request = require('request')
const city = process.argv[2]
const url =
`http://api.weatherapi.com/v1/current.json?key=YOUR_API_KEY&q=${city}`

request({ url, json: true }, (err, res) => {
  if (!err) {
    const d = res.body
    console.log(`City: ${d.location.name}`)
    console.log(`Temp: ${d.current.temp_c}°C`)
    console.log(`Condition: ${d.current.condition.text}`)
  }
})
```

6.

```
const express = require('express')
const app = express()
```

```
app.get('/weather', (req, res) => {
  const city = req.query.city
  res.send({ city, temp: '32', condition: 'Sunny' })
})

app.listen(3000)
```

7.

```
const { MongoClient } = require('mongodb')
const client = new MongoClient('mongodb://localhost:27017')

async function run() {
  await client.connect()
  const db = client.db('taskdb')
  const tasks = db.collection('tasks')
  await tasks.insertMany([
    { description: 'Submit report', completed: false },
    { description: 'Attend meeting', completed: true }
  ])
  const all = await tasks.find().toArray()
  console.log(all)
  await client.close()
}
```

```
run()
```

8.

Note.js

```
const mongoose = require('mongoose')
const Note = mongoose.model('Note', {
  title: String,
  body: String
})
module.exports = Note
```

app.js

```
const express = require('express')
const mongoose = require('mongoose')
const Note = require('./models/note')
const app = express()

mongoose.connect('mongodb://localhost:27017/notesapi')
app.use(express.json())

app.post('/notes', async (req, res) => {
  const note = new Note(req.body)
  await note.save()
  res.send(note)
})

app.get('/notes', async (req, res) => {
  const notes = await Note.find()
  res.send(notes)
})

app.patch('/notes/:id', async (req, res) => {
  const note = await Note.findByIdAndUpdate(req.params.id, req.body, { new: true })
  res.send(note)
})

app.delete('/notes/:id', async (req, res) => {
  await Note.findByIdAndDelete(req.params.id)
  res.send({ status: 'deleted' })
})

app.listen(3000)
```

9.

```
const express = require('express')
const bcrypt = require('bcryptjs')
const jwt = require('jsonwebtoken')
const mongoose = require('mongoose')
```

```
mongoose.connect('mongodb://localhost:27017/authdb')
const User = mongoose.model('User', { email: String, password:
String })

const app = express()
app.use(express.json())

app.post('/users', async (req, res) => {
  const hashed = await bcrypt.hash(req.body.password, 8)
  const user = new User({ email: req.body.email, password:
hashed })
  await user.save()
  res.send(user)
})

app.post('/users/login', async (req, res) => {
  const user = await User.findOne({ email: req.body.email })
  const isMatch = await bcrypt.compare(req.body.password,
user.password)
  const token = jwt.sign({ _id: user._id }, 'secret')
  res.send({ token })
})

const auth = (req, res, next) => {
  const token = req.header('Authorization').replace('Bearer ', '')
  const decoded = jwt.verify(token, 'secret')
  req.user = decoded
  next()
}

app.get('/profile', auth, (req, res) => {
  res.send({ msg: 'Protected route accessed' })
})

app.listen(3000)
```

10.

```
const express = require('express')
const http = require('http')
const socketio = require('socket.io')

const app = express()
const server = http.createServer(app)
const io = socketio(server)

io.on('connection', (socket) => {
  socket.on('message', (msg) => {
    io.emit('message', msg)
  })
})

server.listen(3000)
```

index.html

```
<!DOCTYPE html>
<html>
<body>
  <input id="msg"><button onclick="send()">Send</button>
  <div id="chat"></div>
  <script src="/socket.io/socket.io.js"></script>
  <script>
    const socket = io()
    socket.on('message', m => {
      const d = document.createElement('div')
      d.innerText = m
      document.getElementById('chat').append(d)
    })
    function send() {
      const val = document.getElementById('msg').value
      socket.emit('message', val)
    }
  </script>
</body>
</html>
```