**Task-1**

**Aim:**

Setting up a basic HTTP server: Create a Node.js application that listens for incoming HTTP

requests and responds with a simple message.

**Task-2**

**Aim:**

Experiment with Various HTTP Methods, Content Types and Status Code.

**Task-3**

**Aim:**

Test it using browser, CLI and REST Client.

**Task-4**

**Aim:**

Read File student-data.txt file and find all students whose name contains ‘MA’ and CGPA > 7.

**Task-5**

**Aim:**

Read Employee Information from User and Write Data to file called ‘employee-data.json’

**Task-6**

**Aim:**

Compare Two file and show which file is larger and which lines are different.

**Combined Program for 6 tasks:**

const fs = require("fs");

const url = require("url");

const qs = require("querystring");

const http = require("http");

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

  if (req.method == "GET" && req.url == "/getstudent") {

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

    res.write("Hello, world om!");

    fs.readFile("./student-data.json",(err, data) => {

      if (err) throw err;

      const a = JSON.parse(data);

      let b = [];

      for (const key of a) {

        if (key.name.indexOf("MA") != -1 && key.CGPA > 7) {

          b = b.concat(key.name);

        }

      }

      console.log(b.join(" "));

      res.write(b.join(" "));

      res.end();

    });

    //res.write("Done!")

  } else if(req.method=="GET" && req.url=="/comparefile"){

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

    let file1size = fs.statSync('./first.txt').size;

    let file2size = fs.statSync('./second.txt').size;

    if(file1size > file2size){

      res.write("First file is Larger than second!....</br>")

    }else if(file1size < file2size){

      res.write("Second file is Larger than first!....</br>")

    }else{

      res.write("Both files have same size"+`</br>`)

    }

    const file1 = fs.readFileSync('first.txt',{encoding: 'utf8'}).split('\n');

    const file2 = fs.readFileSync('second.txt',{encoding: 'utf8'}).split('\n');

    for(let i = 0; i <file2.length ; i++){

      if(file1[i] != file2[i]){

        res.write(`Difference at line number ${i+1} </br>`);

      }

    }

    res.end();

  }

  else if (req.method === "POST") {

    res.writeHead(201, { "Content-Type": "text/plain" });

    res.write("Hello, world om!");

    const list = () =>{

      try{

      const readData = fs.readFileSync('employee.json');

      const datajson = JSON.parse(readData)

      return  datajson

      }catch(e){

          return [];

      }

    }

    const urlget = url.parse(req.url);

    const data1 = qs.parse(urlget.query);

    let arr =[];

    arr  = list();

    arr.push({name : data1.name, salary : data1.salary});

    fs.writeFileSync('employee.json',

      JSON.stringify(arr)

    )

    res.end(" post is called");

  } else if (req.method === "PUT") {

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

    res.write("Hello, world om!");

    res.end(" put is called");

  } else if (req.method === "DELETE") {

    res.writeHead(202, { "Content-Type": "text/plain" });

    res.write("Hello, world om!");

    res.end(" delete is called");

  }

});

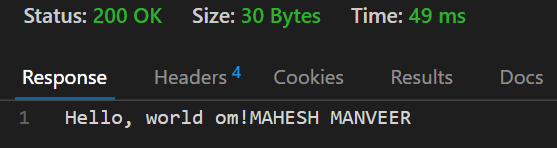
server.listen(8000, () => {

  console.log("listening on port 8000");

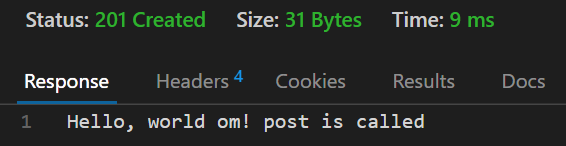
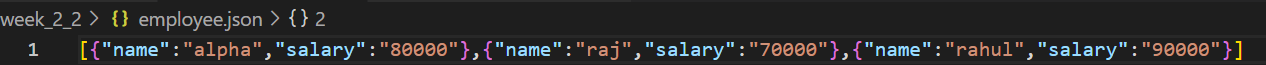
});

**Output:**

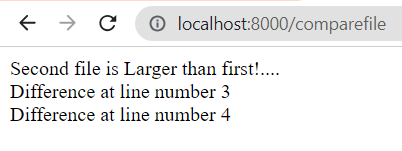
**Task 1,2,3,4 combined output**

****

Task 5

**** **** 

Task 6

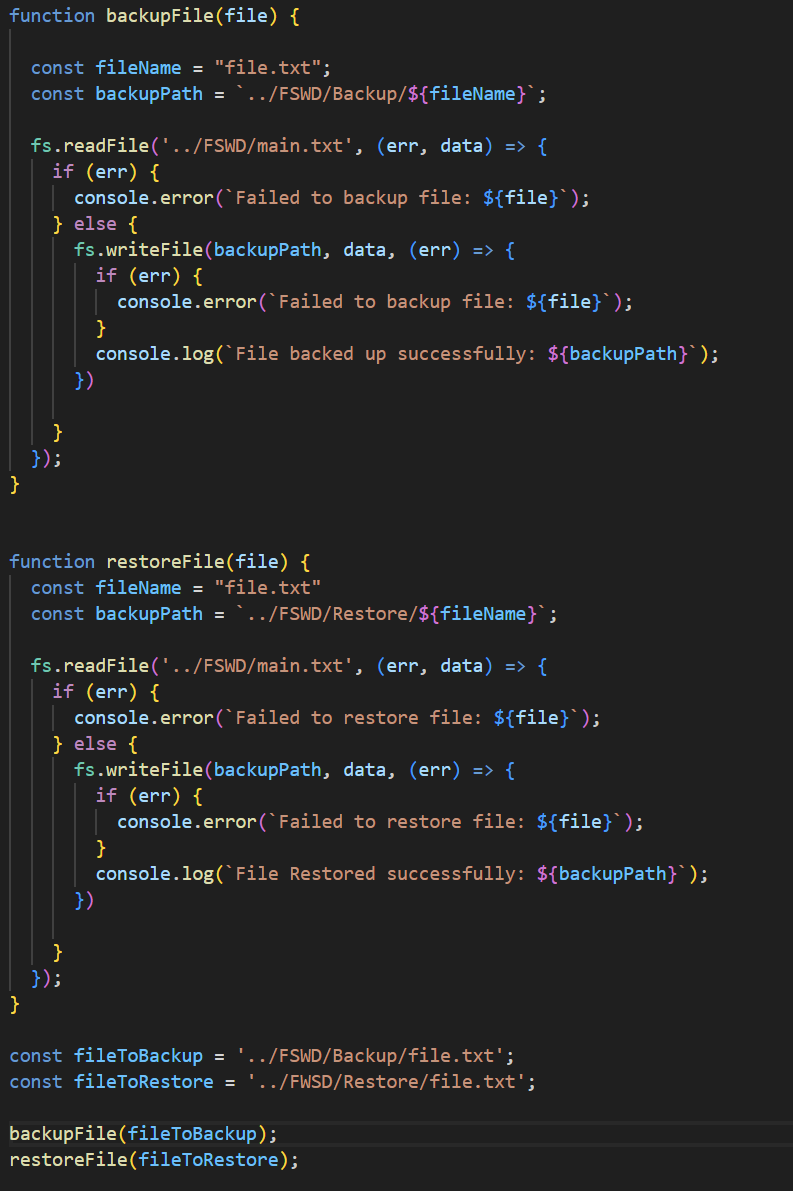


**Task-7**

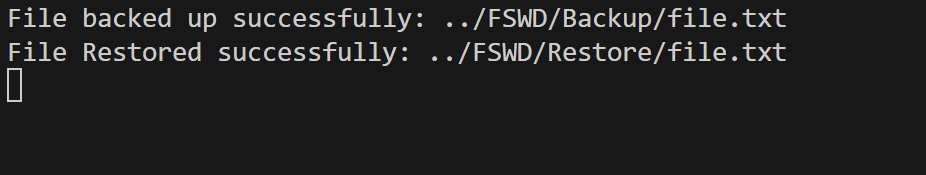
**Aim:**

Create File Backup and Restore Utility.

**Program:**

****

**Output:**

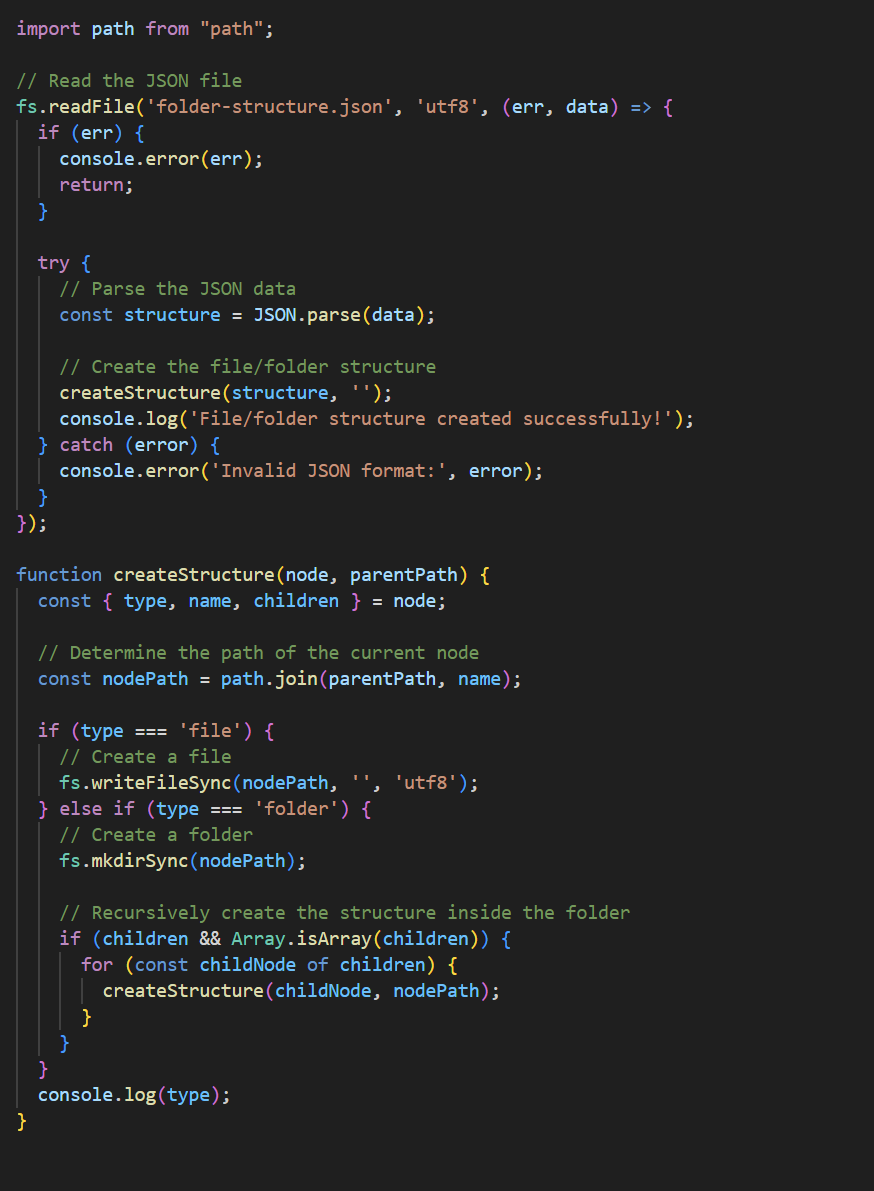


**Task-8**

**Aim:**

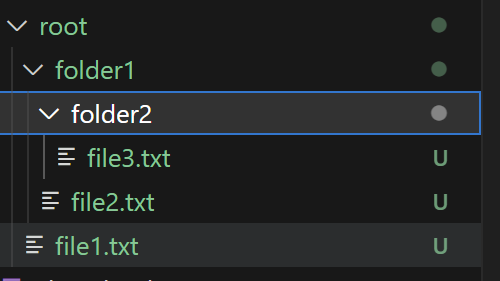
Create File/Folder Structure given in json file.

**Program:**

****

****

**Output:**



**Description:**

fs Module (File System):

The 'fs' module in Node.js provides an API for interacting with the file system. It enables you to perform various file-related operations such as reading from and writing to files, creating and deleting files and directories, modifying file permissions, and more.

http Module:

The 'http' module in Node.js allows you to create HTTP servers and make HTTP requests. It provides a set of functions and classes to handle HTTP-related operations, including creating server instances, sending HTTP requests, and processing incoming HTTP requests.