Node JS Test

Time Limit - 2 hours

Test Guidelines:
All questions are mandatory to attempt
Read the question properly and create a working solution for it.
This test focuses on fundamental concepts of Node and JavaScript.
Create a folder and keep all the files with proper naming .
Push it to github.
Fill out the google form: https://forms.gle/dLdVHLqExgtNEAhk8

Question: Write a JavaScript program that uses a for loop to print numbers from 1 to 10.
Example:
Input: None
Expected Output: 1 2 3 4 5 6 7 8 9 10

```
for (let i = 1; i <= 10; i++) {
  console.log(i);
}
```

Question: Write a JavaScript program that uses a while loop to calculate the sum of numbers from 1 to 10.
Example:
Input: None
Expected Output: 55

```
let s = 0;
let i = 1;
while (i <= 10) {
  s += i;
  i++;
}
console.log( s);
```

Question: Write a JavaScript program that uses a for loop to iterate over the characters in a string and counts the number of vowels.
Example:
Input: "Hello, World!"

Expected Output: 3

```
function vowels(str) {
  const vowels = ['a', 'e', 'i', 'o', 'u'];
  let count = 0;
  for (let i = 0; i < str.length; i++) {
    const char = str[i].toLowerCase();
    if (vowels.includes(char)) {
      count++;
    }
  }
  return count;
}
const ip = "Hello, World!";
const res = vowels(ip);
console.log(res);
```

Question: Write a JavaScript program that uses a while loop to reverse a given number.
Example:
Input: 12345
Expected Output: 54321

```
function rev(n) {
  let res = 0;
  while (n > 0) {
    res = (res * 10) + (n % 10);
    n = Math.floor(n / 10);
  }
  return res;
}
const ip = 12345;
const res = rev(ip);
console.log(res);
```

Question: Write a JavaScript program that checks if a given string is a palindrome.
Example:
Input: "racecar"

Expected Output: true

```
function helper(str) {
    const len = str.length;
    for (let i = 0; i < len / 2; i++) {
        if (str[i] !== str[len - 1 - i]) {
            return false;
        }
    }
    return true;
}
const ip = "racecar";
const res = helper(ip);
console.log(res);
```

Question: Write a Node.js program that creates an HTTP server using the built-in 'http' module and responds with "Hello World!" when accessed.

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello World!');
});

const port = 3000;
server.listen(port, () => {
  console.log(`Server listening on port ${port}`);
});
```

Question: Modify the previous program to include an HTTP header with the content type 'text/html' in the server response.

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/html' });
  res.end('<h1>Hello World!</h1>');
});

const port = 3000;
server.listen(port, () => {
  console.log(`Server listening on port ${port}`);
});
```

Question: Write a Node.js program that creates a server using the built-in 'http' module. Include a custom module called "myDateTime" that returns the current date and time. The server should respond to incoming requests by displaying the current date and time.

```
const http = require('http');
const myDateTime = require('./myDateTime');

const server = http.createServer((req, res) => {
 res.writeHead(200, { 'Content-Type': 'text/plain' });
 res.end(`Current date and time: ${myDateTime.getCurrentDateTime()}`);
});

const port = 3000;
server.listen(port, () => {
 console.log(`Server listening on port ${port}`);
});


exports.getCurrentDateTime = () => {
 const currentDateTime = new Date();
 return currentDateTime.toISOString();
};
```

Question: Write a Node.js program that reads the contents of an HTML file and returns the content as the response when accessed via an HTTP server.

```
const http = require('http');
const fs = require('fs');
const server = http.createServer((req, res) => {
  fs.readFile('index.html', 'utf8', (err, data) => {
    if (err) {
      res.writeHead(500, { 'Content-Type': 'text/plain' });
      res.end('Error reading file');
    } else {
      res.writeHead(200, { 'Content-Type': 'text/html' });
      res.end(data);
    }
  });
});
```

```
const port = 3000;
server.listen(port, () => {
  console.log(`Server listening on port ${port}`);
});
```

Question: Create a Node.js program that appends the text "Hello content!" to a file called "mynewfile1.txt" using the fs.appendFile() method.

```
const fs = require('fs');
const content = 'Hello content!';

fs.appendFile('mynewfile1.txt', content, 'utf8', (err) => {
  if (err) {
    console.error(err);
  } else {
    console.log('Text appended to file successfully!');
  }
});
```

Question: Modify the previous program to create a new file called "mynewfile2.txt" using the fs.open() method with the "w" flag.

const fs = require('fs');

```
const content = 'Hello content!';
const fileName = 'mynewfile2.txt';

fs.open(fileName, 'w', (err, fd) => {
  if (err) {
    console.error(err);
  } else {
    fs.writeFile(fd, content, 'utf8', (err) => {
      if (err) {
        console.error(err);
      } else {
        console.log(`File '${fileName}' created and content written successfully!`);
      }
      fs.close(fd, (err) => {
        if (err) {
          console.error(err);
        }
```

```
      });
    });
  }
});
```

Question: Write a Node.js program that replaces the content of a file called "mynewfile3.txt" with the text "Hello content!" using the fs.writeFile() method.

```
const fs = require('fs');
const content = 'Hello content!';
const fileName = 'mynewfile3.txt';

fs.writeFile(fileName, content, 'utf8', (err) => {
  if (err) {
    console.error(err);
  } else {
    console.log(`Content written to file '${fileName}' successfully!`);
  }
});
```

Question: Implement a Node.js program that appends the text " This is my text." to the end of the file "mynewfile1.txt" using the fs.appendFile() method.

```
const fs = require('fs');
const content = ' This is my text.';
const fileName = 'mynewfile1.txt';

fs.appendFile(fileName, content, 'utf8', (err) => {
  if (err) {
    console.error(err);
  } else {
    console.log(`Text appended to file '${fileName}' successfully!`);
  }
});
```

Question: Develop a Node.js program that deletes a file called "mynewfile2.txt" using the fs.unlink() method.

```
const fs = require('fs');
const fileName = 'mynewfile2.txt';

fs.unlink(fileName, (err) => {
  if (err) {
    console.error(err);
  } else {
    console.log(`File '${fileName}' deleted successfully!`);
  }
});
```

Question: Create a Node.js program that renames a file called "mynewfile1.txt" to "myrenamedfile.txt" using the fs.rename() method.

```
const fs = require('fs');
const oldFileName = 'mynewfile1.txt';
const newFileName = 'myrenamedfile.txt';
fs.rename(oldFileName, newFileName, (err) => {
  if (err) {
    console.error(err);
  } else {
    console.log(`File '${oldFileName}' renamed to '${newFileName}' successfully!`);
  }
});
```

Question: Write a JavaScript function multiplyByTwo that takes a number as an argument and multiplies it by two. Implement a callback function callback that receives the result of the multiplication and displays it on the console.
Example Input: multiplyByTwo(5, callback);
Expected Output:Result: 10

```
function multiplyByTwo(number, callback) {
  const result = number * 2;
```

```
  callback(result);
}

function callback(result) {
  console.log(result);
}

multiplyByTwo(5, callback);
```

Question: Implement a JavaScript function calculateSum that takes an array of numbers as an argument and calculates their sum. The function should accept a callback function callback that receives the calculated sum. Invoke the callback function with the sum of the array elements.
Example Input:calculateSum([2, 4, 6, 8], callback);
Expected Output:Sum: 20

```
function calculateSum(numbers, callback) {
  const sum = numbers.reduce((acc, curr) => acc + curr, 0);
  callback(sum);
}

function callback(sum) {
  console.log("Sum:", sum);
}

calculateSum([2, 4, 6, 8], callback);
```

Question: Create a JavaScript function getUserData that simulates fetching user data from a server asynchronously. The function takes a callback function callback as an argument. Inside the getUserData function, after a delay of 2 seconds, invoke the callback function with a mock user object containing name, email, and age properties.
Example Input: getUserData(callback);

Expected Output:User Data:
{

```
  name: 'John Doe',
  email: 'johndoe@example.com',
  age: 25
}
function getUserData(callback) {
  setTimeout(() => {
    const user = {
      name: 'John Doe',
      email: 'johndoe@example.com',
      age: 25
    };
    callback(user);
  }, 2000);
}

function callback(user) {
  console.log('User Data:');
  console.log(user);
}

getUserData(callback);
```

Question: Write a JavaScript function getRandomNumber that generates a random number between 1 and 10. Implement a promise that resolves with the generated random number. The promise should be rejected if the generated number is less than 5.
Example Input:getRandomNumber()
Expected Output:
A promise object that will be resolved with a random number between 1 and 10 if it is greater than or equal to 5. If the generated number is less than 5, the promise should be rejected.

```
function getRandomNumber() {
  return new Promise((resolve, reject) => {
    const randomNumber = Math.floor(Math.random() * 10) + 1;

    if (randomNumber >= 5) {
      resolve(randomNumber);
    } else {
```

```
      reject('Generated number is less than 5.');
    }
  });
}

getRandomNumber()
  .then((number) => {
    console.log('Resolved:', number);
  })
  .catch((error) => {
    console.error('Rejected:', error);
  });
```

Question: Write a JavaScript function checkFileExists that checks if a file exists asynchronously using promises. The function takes a file path as an argument and returns a promise. Inside the function, after a delay of 1 second, check if the file exists. If the file exists, resolve the promise. If the file does not exist, reject the promise.
Example Input:checkFileExists('/path/to/file.txt')

Expected Output:
A promise object that will be resolved if the file exists after a delay of 1 second. If the file does not exist, the promise should be rejected.

```
function checkFileExists(filePath) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
          const fileExists = true; // Set to true for demonstration purposes

      if (fileExists) {
        resolve('File exists.');
      } else {
        reject('File does not exist.');
      }
    }, 1000);
  });
}
```

```
checkFileExists('/path/to/file.txt')
  .then((message) => {
    console.log('Resolved:', message);
  })
  .catch((error) => {
    console.error('Rejected:', error);
  });
```