Assignment No. 6
EECS 468
Programming Language Paradigms
Due: 11:59 PM, Monday, October 30, 2023
Submit deliverables in a single zip file to Canvas
Files in other formats (e.g., .tar) will not be graded
Name of the zip file: FirstnameLastname_Assignment6 (with your first and last name)
Name of the Assignment folder within the zip file: FirstnameLastname_Assignment6

Deliverables:

1. Copy of Rubric6.docx with your name and ID filled out (do not submit a PDF)
2. JavaScript code for the HTTP server specified below. The Prologue Comments should include the link for replit (as described in the video link below).
3. Steps you followed to test each operation i.e., GET, PUT, DELETE, MKDIR, and Error Handling, the results of the tests, and why you think it your test adequately tested the method.

How to submit the Assignment?

- Please review the student demo video at the link below. It will tell you how you can create, test, and submit your assignment. If you have any questions, please contact one of the GTAs.

https://drive.google.com/drive/folders/1n1R5b3YihQcbCVyQwUjVBD1IMGMdEvBE

Assignment:

- Using what you learned in Chapter 20 about Node.js, write the JavaScript code for an HTTP server that allows remote access to a file system.
- Use the HTTP methods GET, PUT, and DELETE to read, write, and delete files on the server, respectively.
- Then, add support for the MKCOL method ("make collection"), which should create a directory by calling mkdir from the fs module.
- MKCOL is not a widely used HTTP method, but it does exist for this same purpose in the WebDAV standard, which specifies a set of conventions on top of HTTP that make it suitable for creating documents.
- The server should include code to detect an invalid request and return a status code of 405 with the message "The method XXX is not supported.", where XXX is the invalid method requested.
- Write down the steps you followed to test each operation i.e., GET, PUT, DELETE, MKDIR and Error Handling, and the result of the operations. Hint: Use a web browser or curl or postman to send a request. See rubric below.
- Provide comments for the JavaScript code that explain what each line of code is doing. See rubric below.

Rubric for Program Testing		
Exceeds Expectations (90-100%)	**Meets Expectations (80-89%)**	**Unsatisfactory (0-79%)**
Testing procedure is clearly explained for each of the methods: GET, PUT, DELETE, MKDIR, & Error Handling. Sound reasoning on why each test was adequate.	Testing procedure is somewhat explained for each of the methods: GET, PUT, DELETE, MKDIR, & Error Handling. Superficial or unsound reasoning on why each test was adequate.	No testing procedure written for some or all the methods or Error Handling. No reasoning on why some or all of the tests were adequate.

Rubric for Program Comments		
Exceeds Expectations (90-100%)	**Meets Expectations (80-89%)**	**Unsatisfactory (0-79%)**
Software is adequately commented with prologue comments, comments summarizing major blocks of code, and comments on every line.	Prologue comments are present but missing some items or some major blocks of code are not commented or there are inadequate comments on each line.	Prologue comments are missing all together or there are no comments on major blocks of code or there are very few comments on each line.

Adequate Prologue Comments:

- Name of program contained in the file (e.g., EECS 468 Assignment 1)
- Brief description of the program, e.g.,
    - Hello World! examples using JavaScript and HTML
- Inputs (e.g., none, for a function, it would be the parameters passed to it)
- Output, e.g.,
    - Browser window with 2 test buttons
- All collaborators
- Other sources for the code ChatGPT, stackOverflow, etc.
- Author's full name
- Creation date: The date you first create the file, i.e., the date you write this comment

Adequate comments summarizing major blocks of code and comments on every line:

- Provide comments that explain what each line of code is doing.
- You may comment each line of code (e.g., using //) and/or provide a multi-line comment (e.g., using /* and */) that explains what a group of lines does.
- Multi-line comments should be detailed enough that it is clear what each line of code is doing.
- Each block of code must indicate whether you authored the code, you obtained it from one of the sources listed in the prolog, or one of your collaborators authored the code, or if it was a combination of all of these.

Collaboration and other sources for code:

- When you collaborate with other students or use other sources for the code (e.g., ChatGPT, stackOverflow):
    - Your comments must be significantly different from your collaborators.
    - More scrutiny will be applied to grading your comments in particular explaining the code "in your own words", not the source's comments (e.g., ChatGPT's comments).
- Failure to identify collaborators or other sources of code will not only result in a 0 on the assignment but will be considered an act of Academic Misconduct.
- Students who violate conduct policies will be subject to severe penalties, up through and including dismissal from the School of Engineering.