WEB322 Assignment 1

Submission Deadline:

Please check blackboard

Assessment Weight:

5% of your final course Grade

Objective:

In this assignment, students will create a Node.js application that uses the built-in "fs" and "readline" modules to interact with files and directories. This program will read data from a file or directory specified by the user, analyze the content, and generate a report in the console. For submission, students are expected to submit a zipped directory of the code, along with a word/pdf document providing the brief explanation of the code and screenshots of the output.

Step 1: Getting Started (Tools, Files & Directories)

To begin this assignment, make sure you have installed both Visual Studio Code and Node (i.e.: you should be able to run programs written in JavaScript from the Integrated Terminal in Visual Studio Code using the command "node *filename.js*")

- Once you have the above tools installed, create a folder somewhere on your system to store the assignment.
- Download the "files" directory and the "post.txt" file from Blackboard. Unzip the file and place the "files" directory and "post.txt" file within your newly created assignment folder.
- Open Your folder in Visual Studio Code to begin your assignment.
- Create an "a1.js" file (this will be the file that your assignment is written in)
- Finally, your assignment folder should contain the files:

```
(Assignment Folder)
    files
        - apples.txt
        - oranges.txt
        - potatoes.txt
        - zucchini.txt
    a1.js
    post.txt
```

Step 2: User Input

With our files / folders in place, we can begin editing a1.js. The first thing we must do is determine whether the user wishes to analyze a file or directory. This can be done using the "readline" module as mentioned in the notes. The prompts for the user should be the following (sample user responses in **green**):

NOTE: For now, we will simply output the file or directory name to be analyzed with "PENDING".

- Do you wish to process a File (f) or directory (d): **f**
  - File: **post.txt**
    - PENDING: Process file post.txt

- Do you wish to process a File (f) or directory (d): **d**
  - Directory: **files**
    - PENDING: Process directory files

- Do you wish to process a File (f) or directory (d): **abc**
  - Invalid Selection

Step 3: Processing the File

If the user chose the "f" option, then we must process the file that the user entered (ie: "post.txt" from the example above) to generate the following report.

NOTE: If the file cannot be read, output the error to the console using "console.log(err.message);"

Assuming that the following report was generated from the provided "post.txt" file, the user should see the below information in the console (instead of the "PENDING" output created in Step 2):

Number of Characters (including spaces): 2974
Number of Words: 437
Longest Word: Pellentesque

HINTS: To get the contents of the file as a string without any newline characters, the following code may be used:

- .toString().replace(/\s+/g, ' ');

Similarly, to get an array of *words* from the file contents (string), the below code can be used:

- .replace(/[^\w\s\']/g, "").split(' ');

Optional Challenge:

Add the following line to the report: "Most Repeated Word". In the above case, this would be:

Most Repeated Word: amet - 11 times

Step 4: Processing the Directory

If the user chose the "d" option, then we must process the directory that the user entered (ie: "files" from the example above) to generate the following report.

NOTE: If the directory cannot be read, output the error to the console using "console.log(err.message);"

Assuming that the following report was generated from the provided "files", the user should see the below information as a string in the console (instead of the "PENDING" output created in Step 2):

Files (reverse alphabetical order): zucchini.txt, potatoes.txt, oranges.txt, apples.txt

Optional Challenge:

Add the following data to the report for each file: "size" (in bytes). In the above case, this would be:

zucchini.txt: 1361 bytes
potatoes.txt: 1830 bytes
oranges.txt: 1305 bytes
apples.txt: 1512 bytes

Assignment Submission:

- Add the following declaration at the top of your a1.js file

```
/*********************************************************************************
* WEB322 – Assignment 1
* I declare that this assignment is my own work in accordance with Seneca Academic Policy.
* No part of this assignment has been copied manually or electronically from any other source
* (including web sites) or distributed to other students.
*
* Name: _____ Student ID: _____ Date: _____
*
*********************************************************************************/
```

- Compress (.zip) the files in your Visual Studio working directory (this is the folder that you opened in Visual Studio to create your client-side code).
- Submission content includes:
    1. Compressed (.zip) directory.
    2. Word/Pdf document with brief explanation of your code along with the screenshots of your outputs.

Important Note:

- **NO LATE SUBMISSIONS** for assignments. Late assignment submissions will not be accepted and will receive a **grade of zero (0)**.
- Submitted assignments must run locally, i.e.: start up errors causing the assignment/app to fail on startup will result in a **grade of zero (0)** for the assignment.