COMP 3133 – Full Stack Development – Lab 5

- File Module and Processes

Developer Note:

- Try to solve the problems without using search engines or stack overflow for the solutions.
- Create separate files for each exercise

References:

- **https://nodejs.org/api/process.html**
- https://nodejs.org/api/fs.html
- https://nodejs.org/api/path.html
- https://nodejs.org/api/child_process.html

Exercise 1:

- Write a script that lists files in current directory, filtered by a given extension from the command line.
    - Use the **file** and **path module**
    - User the **process object** to get the current directory and command line args
    - Use **fs.readdir(path[, options], callback)** to asynchronously read the contents of the file directory and output the file names

- **Input**

```
Lab5> node ex1.js js
```

➢ **Output**

```
current working directory: C:\COMP3133\_LABS\Solutions\Lab5
command arg - extension : .js
ex1.js
ex2.js
ex3.js
ex4.js
```

Exercise 2:

- Using the **process object** write a script that logs the following https://nodejs.org/api/process.html

    - OS platform
    - processor architecture
    - version of NodeJS
    - version of V8
    - version of LibUV
    - PID
    - title
    - working directory

➢ **Output**

```
== System ==
platform: win32
architecture: x64

== NodeJS ==
node version: 8.11.4
v8 version: 6.2.414.54
libuv version: 1.19.1
== Process ==
pid: 41208
title:
current directory: C:\COMP3133\_LABS\Solutions\Lab5
```

Exercise 3:

- Using following **setInterval** code to run an infinite loop on the process that runs it. Add event listeners on the process **exit** and **signal interrupt** events and log the process running **uptime**. Use Ctrl+X Ctrl+C to kill the current process and trigger the events.

```
setInterval(() => null, 1000)
```

➢ **Output**

```
process uptime on signal interupt 10.955
process uptime on exit: 10.958
```

Exercise 4:

- Given the following **compute.js** file, create a script that will use the **child process module** and **fork a child process** that will execute the compute script and do the following:
    - send a message to the child process to start the calculation
    - create an event listener to receive the message sent from child process and log result

compute.js

```
const longComp = () => {
    let sum = 0;
    for (let i = 0; i < 1e9; i++) {
      sum += i;
    };
    return sum;
  };

  // on message from parent, start long calc
  // send message back to parent when completed
  process.on('message', message => {
    const result = longComp();
    process.send(result);
  });
```

➢ Output

```
Long computation result: 499999999067109000
```

Bonus

- Write a script that reads input from the CLI (command line interface) and logs the uppercased version of the input after Enter is pressed. It should continuously prompt User for an input and end on Ctrl-C + Ctrl-X. Hint, use **process stdout and stdin**

> Output