**Handout-3 Node.js**

**Q1.Run the following codes and analyze the result**

1. var http = require('http');  
   http.createServer(function (req, res) {  
       res.writeHead(200, {'Content-Type': 'text/plain'});  
       res.end('Hello World!');  
   }).listen(8080);

Terminate the program using cntrl+c twice.

1. var http = require('http');  
   http.createServer(function (req, res) {  
       res.writeHead(200, {'Content-Type': 'text/html'});  
       res.end('My World!');  
   }).listen(3000);

Terminate the program using cntrl+c twice.

1. var http = require('http');  
   http.createServer(function (req, res) {  
       res.writeHead(200, {'Content-Type': 'text/html'});  
       res.write("The date and time are good”);  
       res.end();  
   }).listen(8080);

Terminate the program using cntrl+c twice.

**Q2. Execute the following using REPL and analyze the output**

a) $ node

> 1 + 3

4

> 1 + ( 2 \* 3 ) - 4

3

>

b) $ node

> x = 10

10

> var y = 10

undefined

> x + y

c)$ node

> var x = 0

undefined

> do {

... x++;

... console.log("x: " + x);

... } while ( x < 5 );

d) $ node

> var x = 10

undefined

> var y = 20

undefined

> x + y

30

> var sum = \_

undefined

> console.log(sum)

Q3.**Run the following codes and analyze the blocking and non-blocking feature of Node.js**

1. var fs = require("fs");

var data = fs.readFileSync('input.txt');

console.log(data.toString());

console.log("Program Ended");

b)var fs = require("fs");

fs.readFile('input.txt', function (err, data) {

 if (err) return console.error(err);

 console.log(data.toString());

});

console.log("Program Ended");

Q4. **Run the following codes and analyze the blocking( synchronous ) and non-blocking(Asynchronous) feature of Node.js**

var fs = require("fs");

// Asynchronous

read fs.readFile('input.txt', function (err, data) { if (err) { return console.error(err); } console.log("Asynchronous read: " + data.toString()); });

// Synchronous

read var data = fs.readFileSync('input.txt');

console.log("Synchronous read: " + data.toString()); console.log("Program Ended");

**Q5. Execute the code and then note down the path of mynewfile2.txt**

var fs = require('fs');  
 fs.open('mynewfile2.txt', 'w', function (err, file) {  
 if (err) throw err;  
 console.log('Saved!');  
 });

**Q6. Demonstrate the use of** i) “ .load” command to load a previously saved “.js” file

ii) “.save” command to save the current REPL session to a file.

iii) “.help command to display list of all REPL commands”

iv) “ .break” command is used to exit from a multi-line expression

**Q7. Create a script that will read the contents of ./data/exercise-02.txt and output it to console.**

**Q8. Develop a script to display current date and time using your own Date-time Module.**

**Q9. Develop and design a node.js code to display the contents of a file to the client.**

**Q10. Develop and design a node.js code to generate a HTML page to display the text with the h1 and paragraph tags.**

**Q11. Demonstrate the use of addListener(on) and emit() methods of EventEmitter class.**

**Q12.Execute and Analyse the code**

events = require('events');

var eventEmitter = new events.EventEmitter();

var connectHandler = function connected() {

console.log('connection succesful.');

eventEmitter.emit('data\_received');

}

eventEmitter.on('connection', connectHandler);

eventEmitter.on('data\_received', function(){

console.log('data received succesfully.');

});

eventEmitter.emit('connection');

console.log("Program Ended.");

**Q13. Create a new, empty file using the fs.open() method.**

**Q14. i) Create a new file containg hello world message using fs.writeFile() method**

**ii) Replace the contents of a existing “myfile.txt” to “ This is my file”.**

**iii) Rename the “myfile.txt” to “thisfile.txt” using fs.rename()**

**iv) Delete the file “thisfile.txt” using fs.unlink().**

**Q15.Develop and design a node.js code to display the number of lines in a file using split() and length property. Read asynchronously.**

**Q16.Use npm to install a package “upper-case and use it in your code to display a message in uppercase**

**Q17. Demonstrate how more than 1 Handler can be binded to an event using addListener(). Also display the count of handlers attached to an event using listenerCount().**

**Q18.****Develop and design a node.js code to create a readableStream that reads data from a file “myfile.txt” and displays it on your console.**

**Q19.Develop and design a node.js code to create a writablableStream that writes data onto a file “outfile.txt”.**

**Q20. Demonstrate the use of piping streams.**

**Q21.** **Convert the string "abc" into a stream of binary data:**

**Q22. Develop and design a node.js code to create a Buffer with a given string and return the json representation of the buffer.**

**Q23. Demonstrate how to count the number of octets with the use of write () method of buffer class.**

**Q24. Demonstrate the use of length property of buffer class.**

**Q25. Display your current directory and file name.(** **console.log(\_\_dirname);**

**console.log(\_\_filename);**