Unit 2

Q.1) Create an application to demonstrate Node.js Events.

```
var events = require("events");
var em = new events.EventEmitter();
em.on("FirstEvent", function (data) {
  console.log("First subscriber: " + data);
});
em.emit("FirstEvent", "This is my first Node.js event emitter example.");
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL

PS C:\WT> node Tut1.js
First subscriber: This is my first Node.js event emitter example.
PS C:\WT>
```

Custom event :

```
const events = require("events");
const eventEmitter = new events.EventEmitter();
eventEmitter.on("connection", handleConnectionEvent);
eventEmitter.emit("connection");
eventEmitter.emit("connection");
eventEmitter.emit("connection");
eventEmitter.emit("connection");
function handleConnectionEvent() {
  console.log("Conneciton Made!");
}
console.log("End of Program");
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL

PS C:\WT> node Tut1.js
Conneciton Made!
Conneciton Made!
Conneciton Made!
Conneciton Made!
End of Program
PS C:\WT>
```

```
var events = require("events");
var eventEmitter = new events.EventEmitter();
var connectHandler = function connected() {
  console.log("connection successful.");
  eventEmitter.emit("data_recieved");
};
eventEmitter.addListener("connection", connectHandler);
eventEmitter.addListener("data_recieved", function () {
  console.log("data received successfully.");
});
eventEmitter.emit("connection");
console.log("program Ended");
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL

PS C:\WT> node Tut1.js
connection successful.
data received successfully.
program Ended
PS C:\WT>
```

Q.2) Implement all the methods of EventEmitter class.

```
const events = require("events");
const eventEmitter = new events.EventEmitter();
function listner1() {
  console.log("Event recevied by Listner 1");
}
function listner2() {
  console.log("Event recevied by Listner 2");
}
eventEmitter.addListener("write", listner1);
eventEmitter.on("write", listner2);
eventEmitter.emit("write");
console.log(eventEmitter.listenerCount("write"));
eventEmitter.removeListener("write", listner1);
console.log("Listener 1 is removed");
eventEmitter.emit("write");
console.log(eventEmitter.listenerCount("write"));
console.log("Program Ended");
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL

PS C:\WT> node Tut1.js
Event recevied by Listner 1
Event recevied by Listner 2
2
Listener 1 is removed
Event recevied by Listner 2
1
Program Ended
PS C:\WT>
```

Implement Event Emitter Patterns

a) using return value of function

```
var emitter = require("events").EventEmitter;
function LoopProcessor(num) {
  var e = new emitter();
  setTimeout(function () {
    for (var i = 1; i <= num; i++) {
      e.emit("BeforeProcess", i);
      console.log("Processing number:" + i);
      e.emit("AfterProcess", i);
    }
  }, 2000);
  return e;
}
var lp = LoopProcessor(3);
lp.on("BeforeProcess", function (data) {
  console.log("About to start the process for " + data);
});
lp.on("AfterProcess", function (data) {
  console.log("completed processing " + data);
});
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL

PS C:\WT> node Tut1.js
About to start the process for 1
Processing number:1
completed processing 1
About to start the process for 2
Processing number:2
completed processing 2
About to start the process for 3
Processing number:3
completed processing 3
PS C:\WT>
```

b) Extend Event emitter class(using Util module).

```
var emitter = require("events").EventEmitter;
var util = require("util");
function LoopProcessor(num) {
  var me = this;
  setTimeout(function () {
    for (var i = 1; i <= num; i++) {
      me.emit("BeforeProcess", i);
      console.log("Processing Number: " + i);
      me.emit("AfterProcess", i);
    }
  }, 2000);
  return this;
}
util.inherits(LoopProcessor, emitter);
var lp = new LoopProcessor(3);
lp.on("BeforeProcess", function (data) {
  console.log("About to start the process for" + data);
});
lp.on("AfterProcess", function (data) {
  console.log("Completed processing", data);
});
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL

PS C:\WT> node Tut1.js
About to start the process for1
Processing Number: 1
Completed processing 1
About to start the process for2
Processing Number: 2
Completed processing 2
About to start the process for3
Processing Number: 3
Completed processing 3
PS C:\WT>
```

Q.3) Create an application to demonstrate Node.js Functions.

Callback functions

```
/* What is Call back function

A callback is a function passed as an argument to another function.

*/

//callback function - Anonymous Function

const message=function(){

   console.log("Hi I am Bruce Wayne  ");


}

setTimeout(message,3000);

//callback back as an Arrow function

setTimeout(()=>{

   console.log("Calling from arrow funciton");

},3000);'
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL

PS C:\WT> node Tut1.js
Hi I am Bruce Wayne
Calling from arrow funciton
PS C:\WT>
```

```
function displayresult(some)

{

   console.log(some);

}

function calculate(x,y,mycallback)

{

   let sum=x+y;
```

```
    mycallback(sum);
}
calculate(5,10,displayresult);
```

Ouput :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL

PS C:\WT> node Tut1.js
15
PS C:\WT>
```

Standard function :

```
function myfun(num1,num2)
{
    console.log(num1+num2);
    console.log(num1-num2);
    console.log(num1*num2);
    console.log(num1/num2);

}
myfun(12,3);
```

Output :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL

PS C:\WT> node Tut1.js
15
9
36
4
PS C:\WT>
```