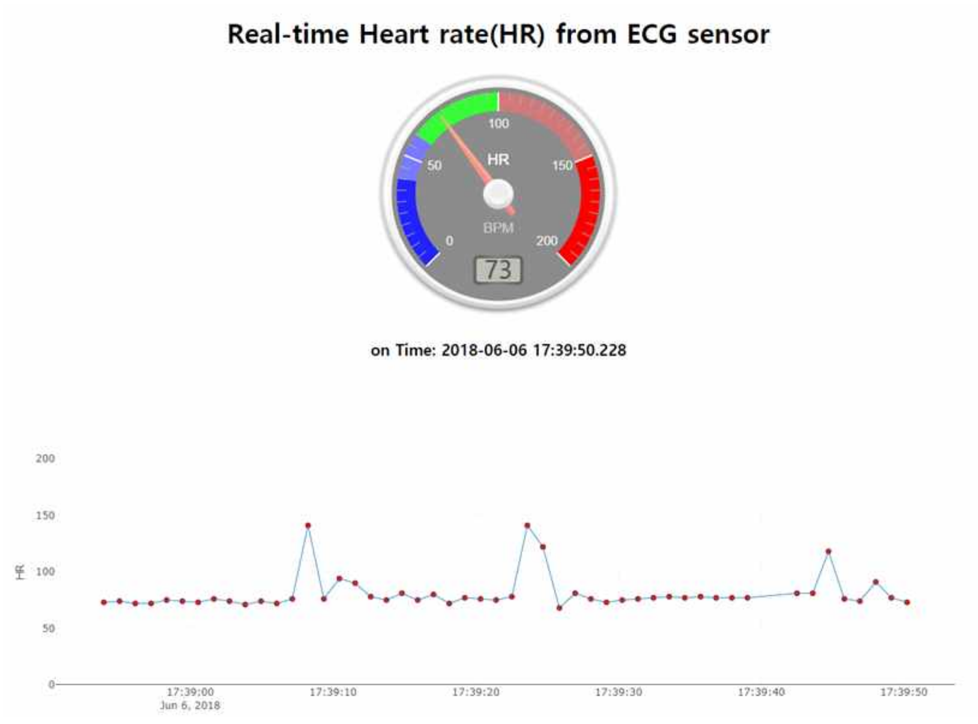
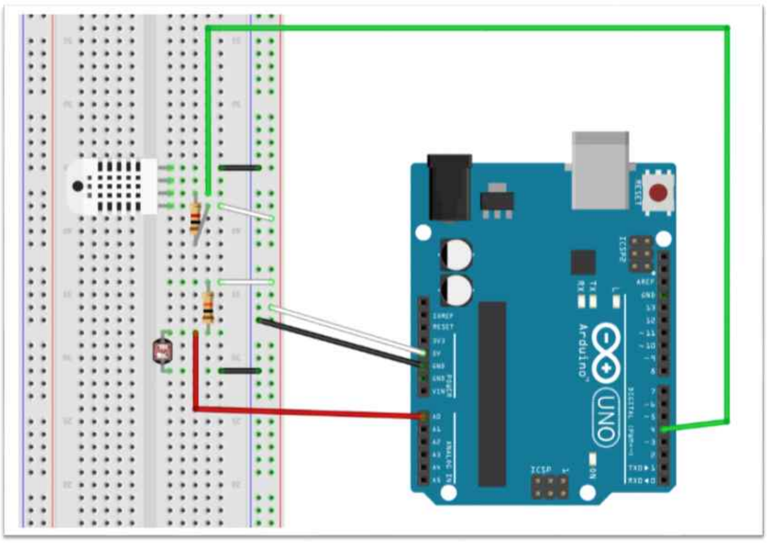

*****   헬스케어신호처리개론: 기말고사   2018.06.14 (목)



1-2. 다음은 CdS, DHT22 센서에서 온도,습도,조도를 측정하여 직렬통신으로 전송하는 아두이노 코드(CdS_DHT22.ino)이다. 밑줄 친 곳에 알맞은 코드는?

```
// CdS + DHT22
#include "DHT.h"
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
#define CDS_INPUT 0
void setup() {
  dht.begin();
  Serial.begin(9600);
}
void loop() {
  int cds_value, lux;
  float temp, humi;
  // Lux from CdS (LDR)
  cds_value = analogRead(CDS_INPUT);
  lux = int(luminosity(cds_value));
  // Reading temperature or humidity takes a given interval!
  // Sensor readings may also be up to 2 seconds 'old'
  humi = [1]___dht.readHumidity();
  // Read temperature as Celsius (the default)
  temp = [1]___dht.readTemperature();
  // Check if any reads failed and exit early (to try again).
  if ([2]___isnan(humi) || isnan(temp) || isnan(lux)) {
    Serial.println("Failed to read from DHT sensor or CdS!");
    return;
  }
  else {
    Serial.print("HS00,");
    Serial.print(temp,1);  // temperature, float
    Serial.print(",");
    Serial.print(humi,1);  // humidity, float
    Serial.print(",");
    Serial.println(lux);    // luminosity, int
  }
  delay(2000);  // 2000 msec, 0.5 Hz
}
//Voltage to Lux
double luminosity (int RawADC0){
  double Vout=RawADC0*5.0/1023.0;  // 5/1023 (Vin = 5 V)
  double lux=(2500/Vout-500)/10;
  return lux;
}
```



1. DHT22 센서에서 습도와 온도를 구하는 객체 변수를 바로 적으시오. ---- (dht)

2. CdS 조도 센서와 DHT22 센서에서 측정한 값이 하나라도 문제가 있는 지를 확인하는 함수는 ?

A. isna(humi) || isna(temp) || isna(lux)

B. isnan(humi) || isnan(temp) || isnan(lux)

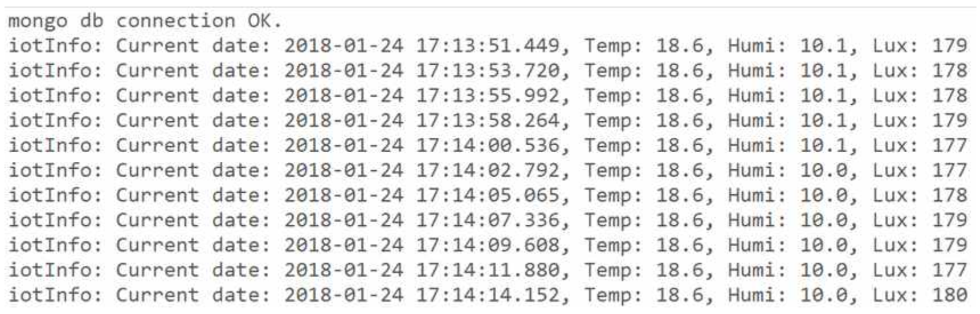
C. isna(humi) && isna(temp) && isna(lux)

D. isnan(humi) && isnan(temp) && isnan(lux)

3-6. 다음은 아두이노에 연결된 CdS, DHT22 센서에서 측정되어 직렬통신으로 전송되는 "ID,온도,습도,조도" 메시지를 처리하여 MongoDB에 저장하는 Nodejs 코드 (cds_dht22_mongodb.js)이다. 밑줄 친 곳에 알맞은 코드는?

```
// cds_dht22_mongodb.js

var serialport = require('serialport');
var portName = 'COM4';  // check your COM port!!
var port     =   process.env.PORT || 3000;

var io = require('socket.io').listen(port);

// MongoDB
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

// MongoDB connection
mongoose.connect('mongodb://localhost:27017/iot');
    var db = mongoose.connection;
    db.on('error', console.error.bind(console, 'connection error:'));
    db.once('open', function callback () {
        console.log("mongo db connection OK.");
});

// Schema
var iotSchema = new Schema({
    date : String,
    temperature : String,
    humidity : String,
    luminosity: String
});

// Display data on console in the case of saving data.
iotSchema.[3]_____methods.info = function () {
    var iotInfo = this.date
    ? "Current date: " + this.date +", Temp: " + this.temperature
    + ", Humi: " + this.humidity + ", Lux: " + this.luminosity
    : "I don't have a date"
    console.log("iotInfo: " + iotInfo);
}

// serial port object
var sp = new serialport(portName,{
    baudRate: 9600,   // 9600  38400  115200
    dataBits: 8,
    parity: 'none',
    stopBits: 1,
    flowControl: false,
    parser: serialport.parsers.readline('₩r₩n')
});
```

```
var readData = '';  // this stores the buffer
var temp ='';
var humi ='';
var lux ='';
var mdata =[]; // this array stores date and data from multiple sensors
var firstcommaidx = 0;
var Sensor = mongoose.model("Sensor", iotSchema);  // sensor data model
sp.on('data', function (data) { // call back when data is received
    readData = data.toString(); // append data to buffer
    firstcommaidx = readData.indexOf(',');
    // parsing data into signals
    if ([4]_____readData.lastIndexOf(',') > firstcommaidx && firstcommaidx > 0) {
        temp = readData.[5]_____substring(firstcommaidx + 1, readData.indexOf(',',firstcommaidx+1));
        humi = readData.substring(readData.indexOf(',',firstcommaidx+1) + 1, readData.lastIndexOf(','));
        lux = readData.substring(readData.lastIndexOf(',')+1);
        readData = '';
        dStr = getDateString();
        mdata[0]=dStr;  // Date
        mdata[1]=temp;  // temperature data
        mdata[2]=humi;  // humidity data
        mdata[3]=lux;   // luminosity data
        var iot = new Sensor({date:dStr, temperature:temp, humidity:humi, luminosity:lux});
        // save iot data to MongoDB
        iot.[6]_____save(function(err, iot) {
            if(err) return handleEvent(err);
            iot.info();  // Display the information of iot data  on console.
        })
        io.sockets.emit('message', mdata);  // send data to all clients
    } else {  // error
        console.log(readData);
    }
});
io.sockets.on('connection', function (socket) {
    // If socket.io receives message from the client browser then this call back will be executed.
    socket.on('message', function (msg) {
        console.log(msg);
    });
    // If a web browser disconnects from Socket.IO then this callback is called.
    socket.on('disconnect', function () {
        console.log('disconnected');
    });
});
// helper function to get a nicely formatted date string
function getDateString() {
    var time = new Date().getTime();
    // 32400000 is (GMT+9 Korea, GimHae)
    // for your timezone just multiply +/-GMT by 3600000
    var datestr = new Date(time +32400000).
    toISOString().replace(/T/, ' ').replace(/Z/, '');
    return datestr;
}
```



```
mongo db connection OK.
iotInfo: Current date: 2018-01-24 17:13:51.449, Temp: 18.6, Humi: 10.1, Lux: 179
iotInfo: Current date: 2018-01-24 17:13:53.720, Temp: 18.6, Humi: 10.1, Lux: 178
iotInfo: Current date: 2018-01-24 17:13:55.992, Temp: 18.6, Humi: 10.1, Lux: 178
iotInfo: Current date: 2018-01-24 17:13:58.264, Temp: 18.6, Humi: 10.1, Lux: 179
iotInfo: Current date: 2018-01-24 17:14:00.536, Temp: 18.6, Humi: 10.1, Lux: 177
iotInfo: Current date: 2018-01-24 17:14:02.792, Temp: 18.6, Humi: 10.0, Lux: 177
iotInfo: Current date: 2018-01-24 17:14:05.065, Temp: 18.6, Humi: 10.0, Lux: 178
iotInfo: Current date: 2018-01-24 17:14:07.336, Temp: 18.6, Humi: 10.0, Lux: 179
iotInfo: Current date: 2018-01-24 17:14:09.608, Temp: 18.6, Humi: 10.0, Lux: 179
iotInfo: Current date: 2018-01-24 17:14:11.880, Temp: 18.6, Humi: 10.0, Lux: 177
iotInfo: Current date: 2018-01-24 17:14:14.152, Temp: 18.6, Humi: 10.0, Lux: 180
```

3. iotSchema 객체에 info() 함수를 추가할 때 사용하는 객체 변수는 ?

A. function B. functions C. method [✓ D. methods]

4. 아두이노에서 전달된 메시지에 담긴 데이터의 유효성을 확인하는 조건문 코드는?

[✓ A. **readData.lastIndexOf(',') > firstcommaidx && firstcommaidx > 0**]

B. **readData.lastIndexOf(',') < firstcommaidx && firstcommaidx > 0**

C. **readData.lastIndexOf(',') > firstcommaidx || firstcommaidx > 0**

D. **readData.lastIndexOf(',') < firstcommaidx || firstcommaidx > 0**

5. “ID,온도,습도,조도” 로 전달되는 메시지에서 온도 값을 읽어내는 코드는?

A. **substring(firstcommaidx, readData.indexOf(',', firstcommaidx))**

B. **substring(firstcommaidx + 1, readData.indexOf(',', firstcommaidx))**

C. **substring(firstcommaidx, readData.indexOf(',', firstcommaidx+1))**

[✓ D. **substring(firstcommaidx + 1, readData.indexOf(',', firstcommaidx+1))**]

6. “ID,온도,습도,조도” 로 전달되는 메시지를 iotSchema 구조를 가진 sensor data model 객체인 iot로 MongoDB에 저장하는 함수는?

A. find B. json [✓ C. save] D. send

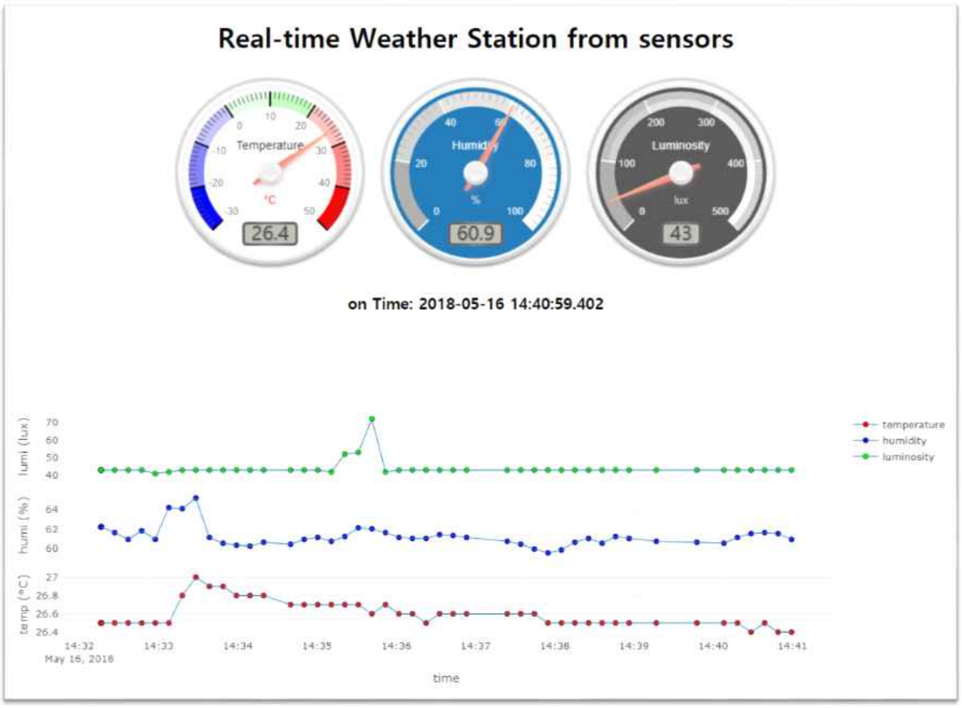
7-10. 다음은 아두이노에 연결된 CdS, DHT22 센서에서 측정되어 직렬통신으로 전송되는 메시지를 Node.js로 처리하여 네트워크 Socket으로 전송되는 데이터를 받아 웹브라우저로 실시간으로 모니터링하는 html 코드 (client_CdS_DHT22.html) 이다.
밑줄 친 곳에 알맞은 코드는?

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>plotly.js Project: Real time signals from multiple sensors</title>
  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
  <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.3.6/socket.io.js"></script>

  <script src="gauge.min.js"></script>

  <style>body{padding:0;margin:30;background:#fff}</style>
</head>

<body>  <!-- style="width:100%;height:100%"> -->
    <!-- Plotly chart will be drawn inside this DIV -->
    <h1 align="center"> Real-time Weather Station from sensors </h1>
    <!-- 1st gauge -->
    <div align="center">
        <canvas id="gauge1"> </canvas>
        <!-- 2nd gauge -->
        <canvas id="gauge2"> </canvas>
        <!-- 3rd gauge -->
        <canvas id="gauge3"> </canvas>
    </div>
    <!-- <div id="console"> </div> -->
    <h3 align="center"> on Time: <span id="time"> </span> </h3>
    <div id="myDiv"></div>
    <hr>

<script>
      /* JAVASCRIPT CODE GOES HERE */
      var streamPlot = document.getElementById('myDiv');
      var ctime = document.getElementById('time');
      var tArray = [], // time of data arrival
        y1Track = [], // value of sensor 1 : temperature
        y2Track = [], // value of sensor 2 : humidity
        y3Track = [], // value of sensor 3 : luminosity
        numPts = 50, // number of data points in x-axis
        dtda = [],  // 1 x 4 array : [date, data1, data2, data3] from sensors
        preX = -1,
        preY = -1,
        preZ = -1,
        initFlag = [7.A]______true;
```

```
var socket = io.connect('http://localhost:3000'); // port = 3000
    socket.on('connect', function () {
        socket.on('message', function (msg) {
            // initial plot
            if(msg[0]!='' && initFlag){
                dtda[0]=msg[0];
                dtda[1]=parseFloat(msg[1]);   // temperature
                dtda[2]=parseFloat(msg[2]);   // Humidity
                dtda[3]=parseInt(msg[3]);   // Luminosity
                init();
                initFlag=[7.B]______false;
            }

            dtda[0]=msg[0];
            dtda[1] = parseFloat(msg[1]);
            dtda[2] = parseFloat(msg[2]);
            dtda[3] = parseInt(msg[3]);

            // Only when any of temperature or Luminosity is different
            // from the previous one, the screen is redrawed.
            if (dtda[1] != preX || dtda[2] != preY || dtda[3] != preZ) {   // any change?
                preX = dtda[1];
                preY = dtda[2];
                preZ = dtda[3];

                // when new data is coming, keep on streaming
                ctime.innerHTML = dtda[0];
                gauge_temp.setValue(dtda[1])   // temp gauge
                gauge_humi.setValue(dtda[2]); // humi gauge
                gauge_lux.setValue(dtda[3]);   // lux gauge

                tArray = tArray.concat(dtda[0]);
                tArray.[8]______splice(0, 1);   // remove the oldest data
                y1Track = y1Track.concat(dtda[1]);
                y1Track.[8]______splice(0, 1); // remove the oldest data
                y2Track = y2Track.concat(dtda[2]);
                y2Track.[8]______splice(0, 1);
                y3Track = y3Track.concat(dtda[3]);
                y3Track.[8]______splice(0, 1);

                var update = {
                    x:  [tArray, tArray, tArray],
                    y:  [y1Track, y2Track, y3Track]
                }

                Plotly.update(streamPlot, update);
            }

      });
    });
```

```
function init() {  // initial screen ()
 // starting point : first data (temp, humi, lux)
 for ( i = 0; i < numPts; i++) {
     tArray.push(dtda[0]);  // date
     y1Track.push(dtda[1]);  // sensor 1 (temp)
     y2Track.push(dtda[2]);  // sensor 2 (humi)
     y3Track.push(dtda[3]);  // sensor 3 (lux)
 }
      Plotly.plot(streamPlot, data, layout);
}

// data
var data = [{
     x : tArray,
     y : y1Track,
     name : 'temperature',
     mode: "markers+lines",
         line: {
             color: "#1f77b4",
             width: 1
         },
         marker: {
             color: "rgb(255, 0, 0)",
             size: 6,
             line: {
               color: "black",
               width: 0.5
             }
         }
  }, {
     x : tArray,
     y : y2Track,
     name : 'humidity',
     xaxis: 'x2',
     yaxis : 'y2',
         mode: "markers+lines",
         line: {
             color: "#1f77b4",
             width: 1
         },
         marker: {
             color: "rgb(0, 0, 255)",
             size: 6,
             line: {
               color: "black",
               width: 0.5
             }
         }
  },
```

```
{
     x : tArray,
     y : y3Track,
     name : 'luminosity',
     xaxis: 'x3',
     yaxis : 'y3',
         mode: "markers+lines",
         line: {
             color: "#1f77b4",
             width: 1
         },
         marker: {
             color: "rgb(0, 255, 0)",  size: 6,
             line: {
                     color: "black",  width: 0.5
                 }
  }}];
// layout
var layout = {
     xaxis : {
         title : 'time',
         domain : [0, 1]
     },
     yaxis : {
         title : 'temp (°C)',
         domain : [0, 0.3],
         range : [-30, 50]
     },
     xaxis2 : {
         title : '',
         domain : [0, 1],
         position : 0.35,
         [9]______showticklabels: false
     },
     yaxis2 : {
         title : 'humi (%)',
         domain : [0.35, 0.65],
         range : [0, 100]
     },
     xaxis3 : {
         title : '',
         domain : [0, 1],
         position : 0.7,
         [9]______showticklabels: false
     },
     yaxis3 : {
         title : 'lumi (lux)',
         domain : [10]______[0.7, 1],
         range : [0, 500]
     }
};
```



7. initFlag의 값으로 각각 알맞은 것은 (A, B 순서대로)?

A. false, false　B. false, true　[✓ C. true, false]　D. true, true

8. 시간 및 센서값 배열에서 가장 오래된 값을 하나 제거하는 코드는?

A. **splice(1)**　[✓ B. **splice(0, 1)**]　C. **split(1)**　D. **split(0, 1)**

9. 위의 실시간 모니터링 그림과 같이 온도 축에만 시간이 표시되고, 습도-, 조도-축에는 시간이 나타나지 않게 하는 설정은?

A. **showticklabel: null**　B. **showticklabels: null**
C. **showticklabel: false**　[✓ D. **showticklabels: false**]

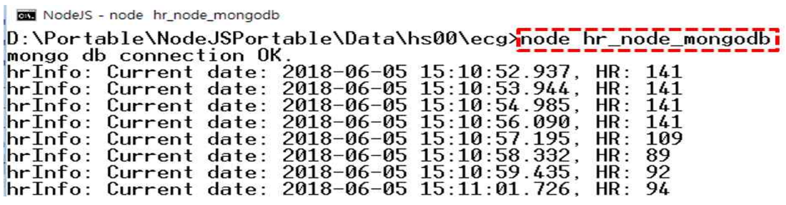
10. 다음 중 조도-축의 y-범위(domain) 설정으로 맞는 것은?

A. **[0.5, 1]**　B. **[0.6, 1]**　[✓ C. **[0.7, 1]**]　D. **[0, 1]**

11-12. 다음은 아두이노에 연결된 SEN0213 심전도 센서에서 측정되어 직렬통신으로 전송되는 "ID,심박수" 메시지를 처리하여 MongoDB에 저장하는 Nodejs 코드 (hr_node_mongodb.js) 이다. 밑줄 친 곳에 알맞은 코드를 바로 적으시오?

```
// hr_node_mongodb.js

var serialport = require('serialport');
var portName = 'COM7';  // check your COM port!!
var port    =    process.env.PORT || 3000;

var io = require('socket.io').listen(port);

// MongoDB
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

// MongoDB connection
mongoose.connect('mongodb://localhost:27017/hrv'); // DB name
    var db = mongoose.connection;
    db.on('error', console.error.bind(console, 'connection error:'));
    db.once('open', function callback () {
        console.log("mongo db connection OK.");
});

// Schema
var hrSchema = new Schema({
    date : String,
    hr : String
});

// Display data on console in the case of saving data.
[11]________hrSchema.methods.info = function () {
    var hrInfo = this.date
    ? "Current date: " + this.date +", HR: " + this.hr
    : "I don't have a date"
    console.log("hrInfo: " + hrInfo);
}

// serial port object
var sp = new serialport(portName,{
    baudRate: 115200,   // 9600  19200  38400
    dataBits: 8,
    parity: 'none',
    stopBits: 1,
    flowControl: false,
    parser: serialport.parsers.readline('₩r₩n')
});
```

```
var readData = ''; // this stores the buffer
var hrv ='';
var mdata =[]; // this array stores date and data from multiple sensors
var firstcommaidx = 0;
var Sensor = mongoose.model("Sensor", hrSchema); // sensor data model
sp.on('data', function (data) { // call back when data is received
    readData = data.toString(); // append data to buffer
    firstcommaidx = readData.indexOf(',');
    // parsing data into signals
    if (firstcommaidx > 0) {
        hrv = readData.substring(firstcommaidx + 1);
        readData = '';
        dStr = getDateString();
        mdata[0]=dStr; // Date
        mdata[1]=hrv; // hr data
        var hrdata = new Sensor({date:dStr, hr:hrv});
        // save data to MongoDB
        [12]________hrdata.save(function(err, iot) {
            if(err) return handleEvent(err);
            hrdata.info(); // Display the information of iot data  on console.
        })
        io.sockets.emit('message', mdata); // send data to all clients
    } else { // error
        console.log(readData);
    }
});
// helper function to get a nicely formatted date string for IOT
function getDateString() {
    var time = new Date().getTime();
    // 32400000 is (GMT+9 Korea, GimHae)
    // for your timezone just multiply +/-GMT by 3600000
    var datestr = new Date(time +32400000).
    toISOString().replace(/T/, ' ').replace(/Z/, '');
    return datestr;
}
io.sockets.on('connection', function (socket) {
    // If socket.io receives message from the client browser then this call back will be executed.
    socket.on('message', function (msg) {
        console.log(msg);
    });
    // If a web browser disconnects from Socket.IO then this callback is called.
    socket.on('disconnect', function () {
        console.log('disconnected');
    });
});
```



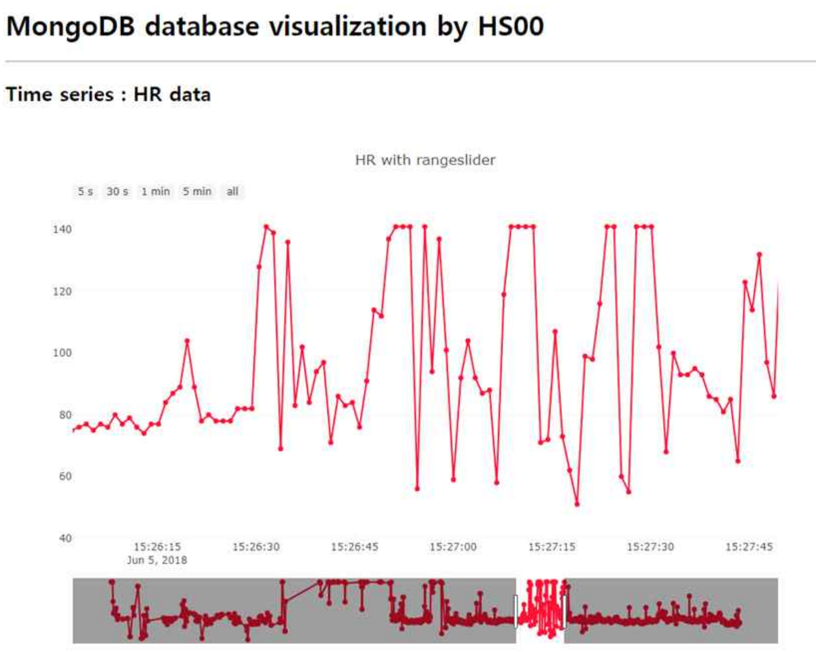
```
D:\Portable\NodeJSPortable\Data\hs00\ecg>node hr_node_mongodb
mongo db connection OK.
hrInfo: Current date: 2018-06-05 15:10:52.937, HR: 141
hrInfo: Current date: 2018-06-05 15:10:53.944, HR: 141
hrInfo: Current date: 2018-06-05 15:10:54.985, HR: 141
hrInfo: Current date: 2018-06-05 15:10:56.090, HR: 141
hrInfo: Current date: 2018-06-05 15:10:57.195, HR: 109
hrInfo: Current date: 2018-06-05 15:10:58.332, HR: 89
hrInfo: Current date: 2018-06-05 15:10:59.435, HR: 92
hrInfo: Current date: 2018-06-05 15:11:01.726, HR: 94
```

13-14. 다음은 MongoDB에 저장된 "ID,심박수" 문서 데이터를 json 파일로 전송하는 라우팅 주소를 지정하는 'express' 웹서버를 구동하는 Nodejs 코드 (hr_express.js) 이다.
밑줄 친 곳에 알맞은 코드를 바로 적으시오.

```
// hr_express.js

// Express with CORS
var express = require('express');
var cors = require('cors');  // CORS: Cross Origin Resource Sharing
var app = express();
app.use(cors()); // CORS
var web_port = 3030;  // express port

// MongoDB
var mongoose = require('mongoose');
var Schema = mongoose.Schema;  // Schema object
// MongoDB connection
mongoose.connect('mongodb://localhost:27017/hrv'); // DB name
var db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function callback () {
        console.log("mongo db connection OK.");
});
// Schema
var hrSchema = new Schema({
    date : String,
    hr : String
});
var Sensor = mongoose.model("Sensor", hrSchema);  // sensor data model

// Web routing address
app.get('/', function (req, res) {  // localhost:3030/
  res.send('Hello Arduino-HR IOT!');
});
// find all data & return them
app.get('/hrv', function (req, res) {
    [13]_______Sensor.find(function(err, data) {
        res.[14]_______json(data);
    });
});
// find data by id
app.get('/hrv/:id', function (req, res) {
    [13]_______Sensor.findById(req.params.id, function(err, data) {
        res.[14]_______json(data);
    });
});

// Express WEB
app.use(express.static(__dirname + '/public'));  // WEB root folder
app.listen(web_port);  // port 3030
console.log("Express_HR_IOT is running at port:3030");
```

15-16. 다음은 MongoDB에 저장된 “ID,심박수” 문서 데이터를 json 파일로 반환해주는 라우팅주소로 Node express 서버에 접속하는 웹클라이언트 html 코드 (client_hrDB.html) 이다.
밑줄 친 곳에 알맞은 코드는?

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <!-- Plotly.js -->
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
<body>
    <h1>MongoDB database visualization by HS00</h1>
    <hr>
    <h2>Time series : Heart rate</h2>

    <!-- Plotly chart will be drawn inside this DIV -->
    <div id="myDiv" style="width: 1000px;height: 700px"></div>

    <script>
        <!-- JAVASCRIPT CODE GOES HERE -->

        Plotly.d3.json("http://localhost:3030/hrv", function(err, json){
            //alert(JSON.stringify(json));  // It works!!!

            if(err) throw err;

            var date = [];
            var hrv = [];
            var jsonData = eval(JSON.stringify(json));

            for (var i = 0; i < jsonData.length; i++) {
                date[i] = jsonData[i].date;
                hrv[i] = jsonData[i].hr;
            }
            // time series of sensor data
            var trace1 = {
                type: "scatter",
                mode: "lines+markers",
                name: 'Heart rate',
                x: date,
                y: [15]_______hrv,
                line: {color: '#fc1234'}
            }

            var data = [trace1];
    // Layout with builtin rangeslider
            var layout = {
                title: 'HR with rangeslider',
                xaxis: {
                    autorange: true,
                    range: [date[0], date[[16]_______date.length-1]],
                    rangeselector: {buttons: [
                        {
                            count: 5,
                            label: '5 s',
                            step: 'second',
                            stepmode: 'backward'
                        },
                        {
                            count: 30,
                            label: '30 s',
                            step: 'second',
                            stepmode: 'backward'
                        },
                        {
                            count: 1,
                            label: '1 min',
                            step: 'minute',
                            stepmode: 'backward'
                        },
                        {
                            count: 5,
                            label: '5 min',
                            step: 'minute',
                            stepmode: 'backward'
                        },
                        {step: 'all'}
                        ]},
                        rangeslider: {range: [date[0], date[[16]_______date.length-1]]},
                        type: 'date'
                    },
                    yaxis: {
                        autorange: true,
                        range: [0, 200],
                        type: 'linear'
                    }
                };

                Plotly.newPlot('myDiv', data, layout);
            })

        </script>

    </body>
</html>
```


MongoDB database visualization by HS00

Time series : HR data

15. y-축에 그려질 심박수 정보를 담고 있는 변수는?

A. hr [✓ B. hrv] C. jsonData D. date

16. **rangeslider의 x-축에 전체 데이터의 시간 범위를 지정하는 코든는?**

A. **data.length** B. **data.length-1**

C. **date.length** [✓ D. **date.length-1**]

17. 다음 중 NoSQL 문서 데이터베이스인 MongoDB의 기본 구성 요소가 아닌 것은?

A. document [✓ B. table] C. collection D. database

18. 문서명이 'sensor'인 MongoDB에서 가장 최근 문서 100개를 추출하는 명령문은?

A. db.sensor.find().sort({_id: 1}).limit(100)

B. db.sensors.find().sort({_id: 1}).limit(100)

C. db.sensor.find().sort({_id: -1}).limit(100)

[✓ D. db.sensors.find().sort({_id: -1}).limit(100)]

19. id가 'hs99'인 친구의 심박변이가 담긴 csv 파일 (hs99hr.csv)을 나의 MongoDB에 새로운 DB로 저장하는 명령은?

[✓ A. mongoimport -d hs99 -c sensors --type csv --headerline —file hs99hr.csv]

B. mongoimport -d hs99 -c sensors --type csv —file hs99hr.csv

C. mongoimport -d hs99 -s sensors --type csv --headerline —file hs99hr.csv

D. mongoimport -d hs99 -s sensors --type csv —file hs99hr.csv

20. 문제 11번의 Node 코드인 hr_node_mongodb.js로 MongoDB에 저장된 'hrv' 데이터베이스에서 최근 문서 500개를 추출해서 'hr500.csv'로 저장하는 명령은?

A. mongoexport -d hrv -c sensors --sort "{_id: 1 }" --limit=500 --fields date,hr —type=csv --out hr500.csv

B. mongoexport -d hrv -c sensors --sort "{_id: 1 }" --limit=500 --fields date,hrv —type=csv --out hr500.csv

[✓ C. mongoexport -d hrv -c sensors --sort "{_id: -1 }" --limit=500 --fields date,hr —type=csv --out hr500.csv]

D. mongoexport -d hrv -c sensors --sort "{_id: -1 }" --limit=500 --fields date,hrv —type=csv --out hr500.csv