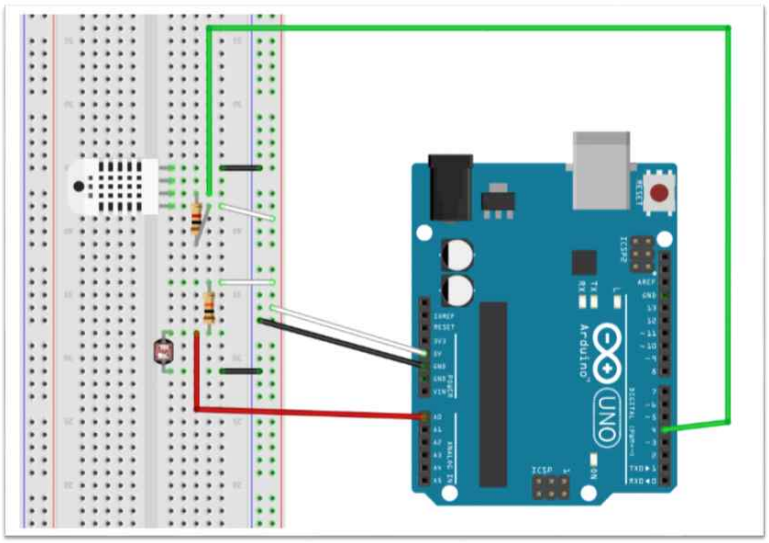

***** 아두이노응용: 기말고사 2018.12.11 (화)

1-2. 다음은 CdS, DHT22 센서에서 온도,습도,조도를 측정하여 직렬통신으로 전송하는 아두이노 코드(AAnn_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 = [1]_____int(luminosity(cds_value));
  // Reading temperature or humidity takes a given interval!
  // Sensor readings may also be up to 2 seconds 'old'
  humi = dht.readHumidity();
  // Read temperature as Celsius (the default)
  temp = 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("AA00,");
    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. CdS 센서에서 조도를 정수로 구하는 코드는? ---- (int)

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

A. isnan(humi) && isnaa(temp) && isnan(lux)

B. isnull(humi) && isnull(temp) isnull(lux)

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

D. isnull(humi) || isnull(temp) || isnull(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.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: " + [3]_____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 (readData.lastIndexOf(',') > firstcommaidx && firstcommaidx > 0) {
        temp = readData.[4]_____substring(firstcommaidx + 1, readData.indexOf(',',firstcommaidx+1));
        humi = readData.substring(readData.indexOf(',',firstcommaidx+1) + 1, readData.lastIndexOf(','));
        lux = readData.[5]_____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. 위와 같은 Node 실행 결과를 얻기 위한 코드는 ?

--- (iotInfo)

4. “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))

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

A. substring(readData.lastIndexOf(','))
B. substring(readData.lastIndexOf(',')+1)
C. substring(readData.endIndexOf(','))
D. substring(readData.endIndexOf(',')+1)

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

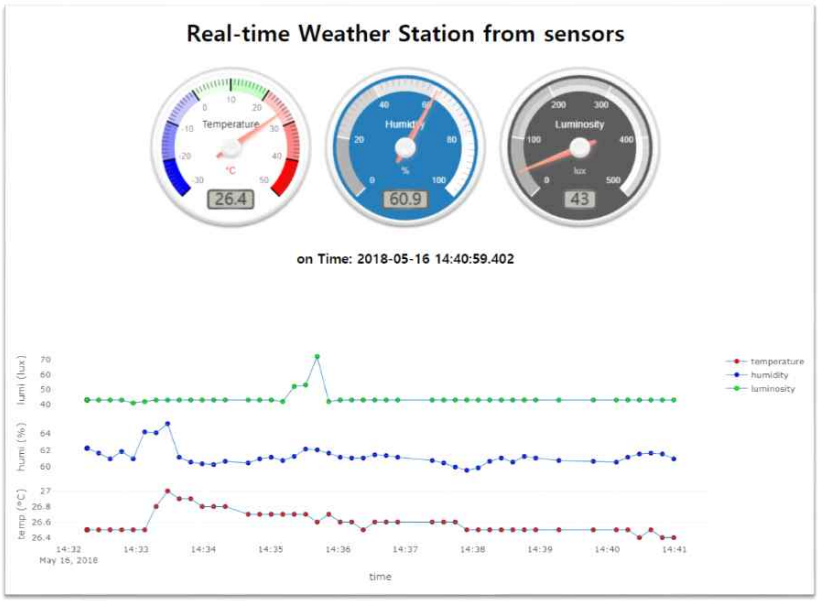
A. find B. json C. save D. send

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 = true;
```

```
var socket = io.connect('http://localhost:3000'); // port = 3000
    socket.on('connect', function () {
        socket.on('message', function (msg) {
            // initial plot
            if([7]_____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=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); // remove the oldest data
                y3Track = y3Track.concat(dtda[3]);
                y3Track.[8]_____splice(0, 1); // remove the oldest data

                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. 메시지(message)가 소켓으로 정상적으로 처음 들어올 때 init() 함수를 한번 실행하기 위한 조건으로 알맞은 것은?

A. msg[0] != '' && initFlag
B. msg[0] != '' || initFlag
C. msg[0] = '' && initFlag
D. msg[0] = '' || initFlag

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: false
B. showticklabel: true
C. showticklabels: false
D. showticklabels: true

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

A. [0.7, 1]
B. [0.8, 1]
C. [0.9, 1]
D. [0, 1]

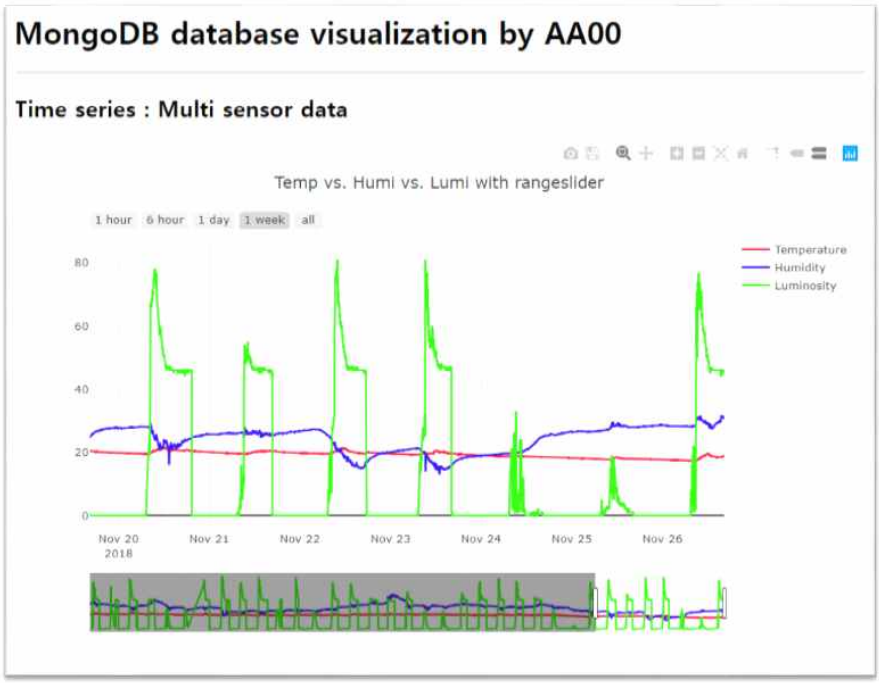
11-13. 다음은 MongoDB에 'iot' 로 저장된 "_id,날짜,온도,습도,조도" 문서 데이터를 json 파일로 전송하는 라우팅 주소를 지정하는 'express' 웹서버를 구동하는 Nodejs 코드 (cds_dht22_express.js) 이다. 밑줄 친 곳에 알맞은 코드를 바로 적으시오.

```
// cds_dht22_express.js  // Express with CORS
var express = require('express');
var cors = require('cors');  // CORS: Cross Origin Resource Sharing
var app = express();
// CORS
app.use([11]_____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/iot'); // 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 iotSchema = new Schema({
    date : String,
    temperature : String,
    humidity : String,
    luminosity: String
});
var Sensor = mongoose.model("Sensor", iotSchema);  // sensor data model

// Web routing address
app.get('/', function (req, res) {  // localhost:3030/
  res.send('Hello Arduino IOT!');
});
// find all data & return them
app.get('/iot', function (req, res) {
    [12]_____Sensor.find(function(err, data) {
        res.[13]_____json(data);
    });
});
// find data by id
app.get('/iot/:id', function (req, res) {
    [12]_____Sensor.findById(req.params.id, function(err, data) {
        res.[13]_____json(data);
    });
});
// Express WEB
app.use(express.static(__dirname + '/public'));  // WEB root folder
app.listen(web_port);  // port 3030
console.log("Express_IOT is running at port:3030 with CORS powered!");
```

14-16. 다음은 MongoDB에 저장된 "_id,날짜,온도,습도,조도" 문서 데이터를 json 파일로 반환해주는 라우팅주소로 Node express 서버에 접속하는 웹클라이언트 html 코드 (client_iotDB.html) 이다. 밑줄 친 곳에 알맞은 코드는?

```
<!DOCTYPE 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 AA00</h1>
        <hr>
        <h2>Time series : Multi sensor data</h2>

        <!-- Plotly chart will be drawn inside this DIV -->
        <div id="myDiv" style="width: 900px;height: 600px"></div>

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

                Plotly.d3.json("http://localhost:3030/[14]_____iot", function(err, json){
                        if(err) throw err;

                        var date = [];
                        var temp = [];
                        var humi = [];
                        var lumi = [];
                        var jsonData = [15]_____eval(JSON.stringify(json));

                        for (var i = 0; i < jsonData.length; i++) {
                                date[i] = jsonData[i].date;
                                temp[i] = jsonData[i].temperature ;
                                humi[i] = jsonData[i].humidity;
                                lumi[i] = jsonData[i].luminosity;

                        }

                        var trace1 = {
                                type: "scatter",
                                mode: "lines",
                                name: 'Temperature',
                                x: date,
                                y: temp,
                                line: {color: '#fc1234'}
                        }
```

```
var trace2 = {
        type: "scatter", mode: "lines",
        name: 'Humidity',
        x: date,
        y: humi,
        line: {color: '#3412fc'}
}
var trace3 = {
        type: "scatter", mode: "lines",
        name: 'Luminosity',
        x: date,
        y: lumi,
        line: {color: '#34fc12'}
}
var data = [trace1, trace2, trace3];
var layout = {
        title: 'Temp vs. Humi vs. Lumi with rangeslider',
        xaxis: {
                autorange: true,
                range: [[16]_____date[0], date[date.length-1]],
                rangeselector: {buttons: [
                        {
                                count: 1, label: '1 hour',
                                step: 'hour',
                                stepmode: 'backward'
                        },
                        {
                                count: 6, label: '6 hour',
                                step: 'hour',
                                stepmode: 'backward'
                        },
                        {
                                count: 24, label: '1 day',
                                step: 'hour',
                                stepmode: 'backward'
                        },
                        {
                                count: 7, label: '1 week',
                                step: 'day',
                                stepmode: 'backward'
                        },
                        {step: 'all'}
                        ]},
        rangeslider: {range: [[16]_____date[0], date[date.length-1]]},
                        type: 'date'
                },
                yaxis: {
                        autorange: true,
                        range: [0, 300],
                        type: 'linear' }
        };
```



14. MongoDB에 저장된 "_id,날짜,온도,습도,조도" 문서 데이터를 json 파일로 반환해주는 웹라우팅주소는 무엇인가?

--- (iot)

15. 웹에서 MongoDB에 접속해서 받은 json 파일을 문자열 객체로 만드는 코드는?

A. eval(JSON.stringify(json))
B. get(JSON.stringify(json))
C. JSON.stringify(json)
D. eval(JSON.substring(json))

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

A. date[0], date[date.length]
B. date[0], date[date.length-1]
C. date[1], date[date.length]
D. date[1], date[date.length-1]

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

A. schema B. document C. collection D. database

18. 문서명이 'sensor'인 MongoDB에서 가장 최근 문서 10개를 추출하는 명령문은?

A. db.sensor.find().sort({_id: 1}).limit(10)
B. db.sensors.find().sort({_id: 1}).limit(10)
C. db.sensor.find().sort({_id: -1}).limit(10)
D. db.sensors.find().sort({_id: -1}).limit(10)

19. id가 'aa99'인 친구의 센서데이터가 담긴 csv 파일 (aa99.csv)을 나의 MongoDB에 새로운 DB 'aa77iot'로 저장하는 명령은?

A. mongoexport -d aa77iot -c sensors --type csv --headerline —file aa99.csv
B. mongoimport -d aa77iot -c sensors --type csv —headerline —file aa99.csv
C. mongoexport -d aa77iot -s sensors --type csv --headerline —file aa99.csv
D. mongoimport -d aa77iot -s sensors --type csv —headerline —file aa99.csv

20. 문제 [3~6]번의 Node 코드인 cds_dht22_mongodb.js로 MongoDB에 저장된 'iot' 데이터베이스에서 최근 문서 500개를 추출해서 'iot500.csv'로 저장하는 명령은?

A. mongoexport –d iot –c sensors --sort "{_id: 1 }" --limit=500 —fields date, temperature,humidity,luminosity —type=csv —out iot500.csv

B. mongoexport –d iot –c sensor --sort "{_id: 1 }" --limit=500 —fields date, temperature,humidity,luminosity —type=csv —out iot500.csv

C. mongoexport –d iot –c sensors --sort "{_id: -1 }" --limit=500 —fields date, temperature,humidity,luminosity —type=csv —out iot500.csv

D. mongoexport –d iot –c sensor --sort "{_id: -1 }" --limit=500 —fields date, temperature,humidity,luminosity —type=csv --out iot500.csv