Q1)

a)Write a javascript to convert to and from celcius, farenheit.

AIM:

ALGORITHM:

CODE:

*//Write a javascript to convert to and from celcius, farenheit prompt the user for input and output the result to the console.*

*// [ Formula : c/5 = (f-32)/9 [ where c = temperature in celcius and f = temperature in farenheit ]*

const prompt = require('prompt-sync')();

var c = prompt("Enter the temperature in celcius");

var f = prompt("Enter the temperature in farenheit");

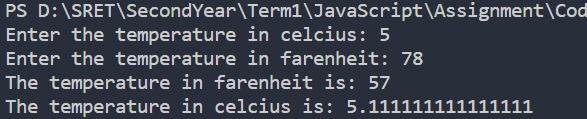
var celcius = (f-32)/9;

var farenheit = (c\*5)+32;

console.log("The temperature in farenheit is "+farenheit);

console.log("The temperature in celcius is "+celcius);

OUTPUT:



b)Write a javascript to determine whether a given year is a leap year in the Georgian calender

AIM:

ALGORITHM:

CODE:

*//Write a javascript to determine whether a given year is a leap year in the Georgian calender*

const prompt = require('prompt-sync')();

var year = prompt("Enter the year: ");

if(year%4==0&&year%100!=0||year%400==0){

    console.log("It is a leap year");

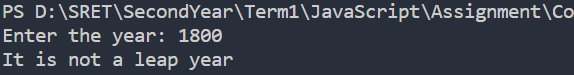
}

else{

    console.log("It is not a leap year");

}

OUTPUT:



c)Write a javascript to display square and cube of a number from 1 to 10 in web browser.

AIM:

ALGORITHM:

CODE:

index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Square and cube</title>

</head>

<body>

    <table border = 1 align="centre">

        <th>Number</th>

        <th>Square</th>

        <th>Cube</th>

        <script type="text/javascript" src="index.js"></script>

    </table>

</body>

</html>

index.js

for(let i=1;i<=10;i++){

    document.write(

        "<tr><td>"+

        i+

        "</td><td>"+

        i\*i+

        "</td><td>"+

        i\*i\*i+

        "</td></tr>"

    );

}

Q2)

1. Write a javascript to display whether a number is prime or not in alert box.

AIM:

ALGORITHM:

CODE:

<!DOCTYPE html>

<html>

<head>

    <title>Q2a</title>

</head>

<body>

    <script type="text/javascript">

        var num = prompt("Enter a number");

        var flag = 0;

        for(var i=2; i<num; i++){

            if(num%i == 0){

                flag = 1;

                break;

            }

        }

        if(flag == 0){

            alert(num + " is a prime number");

        }

        else{

            alert(num + " is not a prime number");

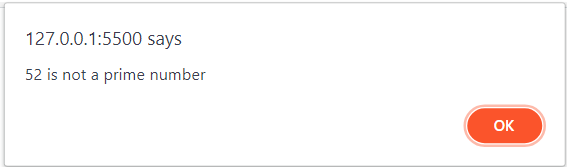
        }

    </script>

</body>

</html>

OUTPUT:



1. Write a javascript for performing calculator operations(+,-,\*,/,%) using switch case.

AIM:

ALGORITHM:

CODE:

const prompt = require('prompt-sync')();

var num1 = prompt("Enter the first number: ");

var num2 = prompt("Enter the second number: ");

var operator = prompt("Enter the operator: ");

switch(operator){

    case '+':

        console.log("The sum is: "+(num1+num2));

        break;

    case '-':

        console.log("The difference is: "+(num1-num2));

        break;

    case '\*':

        console.log("The product is: "+(num1\*num2));

        break;

    case '/':

        console.log("The quotient is: "+(num1/num2));

        break;

    case '%':

        console.log("The remainder is: "+(num1%num2));

        break;

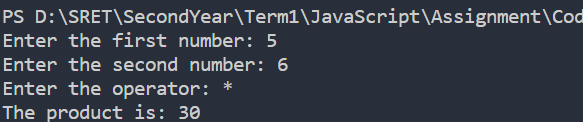
    default:

        console.log("Invalid operator");

        break;

}

OUTPUT:



Q3)

a)Write a javascript to find sum of elements in array using function (prompt the inputs from user)

AIM:

ALGORITHM:

CODE:

const prompt = require('prompt-sync')();

var arr = [];

var n = parseInt(prompt("Enter the number of elements: "));

for(var i=0;i<n;i++){

    arr[i] = parseInt(prompt("Enter the element: "));

}

var sum = 0;

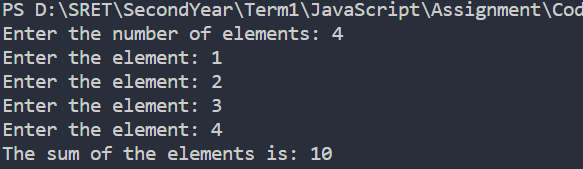
for(var i=0;i<n;i++){

    sum = sum + arr[i];

}

console.log("The sum of the elements is: "+sum);

OUTPUT:



b)Write a recursive function in javascript to print fibonacci series (prompt input from the user)

AIM:

ALGORITHM:

CODE:

const prompt = require('prompt-sync')();

var n = parseInt(prompt("Enter the number of terms: "));

var a = 0;

var b = 1;

var c;

console.log(a);

function fibo(n){

    if(n>0){

        c = a+b;

        a = b;

        b = c;

        console.log(a);

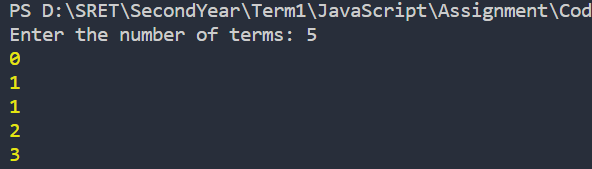
        fibo(n-1);

    }

}

fibo(n-1);

OUTPUT:



Q4)

Create an object called 'person' with the following properties

• 'name' (string)

•'age' (number)

• 'gender' (String)

• 'isStudent' (Boolean)

• Create a method called 'greet' for the 'person' object that logs a greeting message to the console, including the person's name and age

Access the 'name' property of the 'person' object and store it in a variable called 'personName'

ii) Add a new property 'city' to the 'person' object and set its value to a city of your choice. Also, modify the 'age' property to increase it by 5

iii) Loop through the 'person' object and log each property and its value to the console

iv) Create another object called 'address' as a nested object within 'person'

with the following properties

'street' (string)

'postalCode' (string)

Access the 'postalCode' property inside the 'address' object and store it in a variable called 'postC

AIM:

ALGORITHM:

CODE:

var person = {

        name: "Sai",

        age: 20,

        gender: ‘Male’,

        isStudent: true,

        greet: function(){

            console.log("Hello "+this.name+" your age is "+this.age);}

    };

    var personName = person.name;

    person.city = "Hyderabad";

    person.age = person.age + 5;

    for(var key in person){

        console.log(key+" : "+person[key]);

    }

    var address = {

        street: "xyz",

        postalCode: "123"

    };

    var postC = address.postalCode;

    console.log(postC);

    person.address = address;

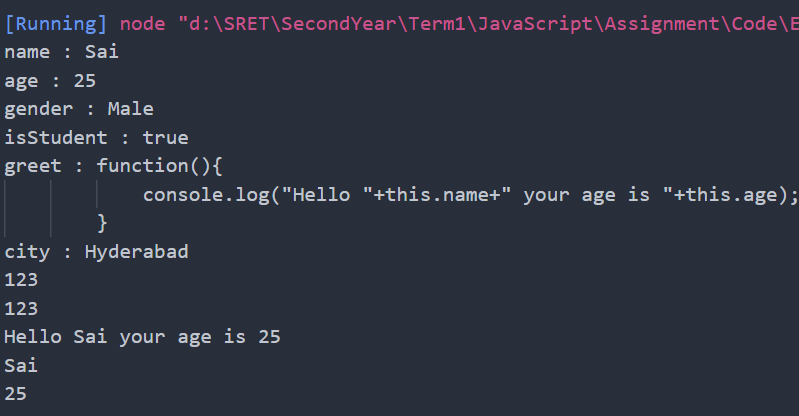
    console.log(person.address.postalCode);

    person.greet();

    console.log(personName);

    console.log(person.age);

OUTPUT:

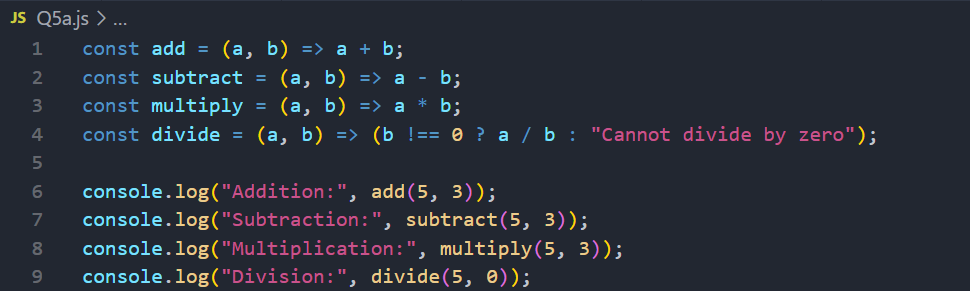


Q5)A) Designing calculator using Arrow functions in JavaScript

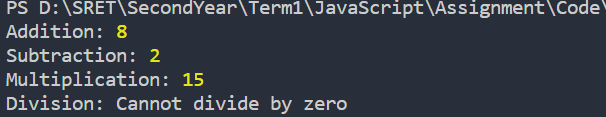
AIM:

ALGORITHM:

Code:



OUTPUT:



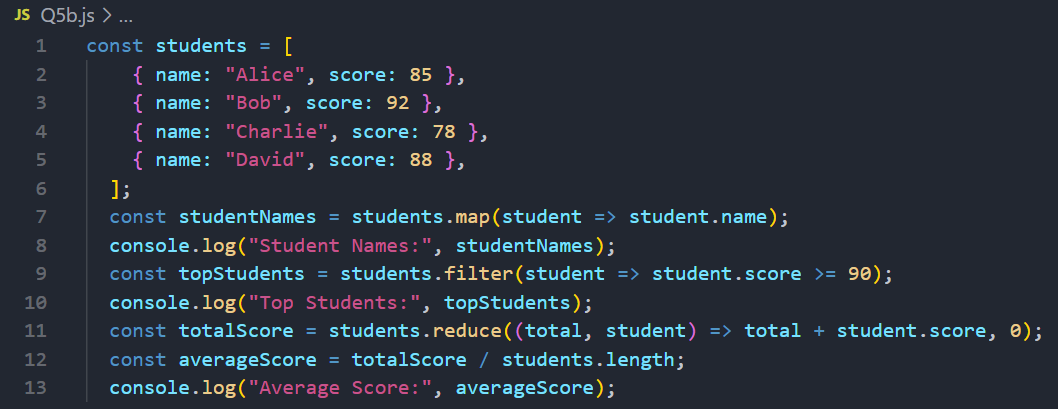
Q5)B)

Illustrating array helper methods in array of objects

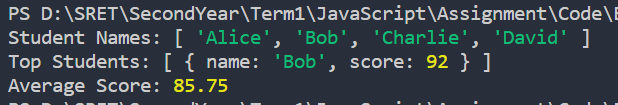
AIM:

ALGORITHM:

CODE:



OUTPUT:



Q6) Illustration of Enhanced Object Literal

AIM:

ALGORITHM:

CODE:



OUTPUT:



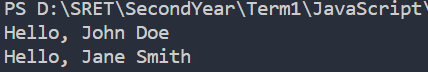
Q7) Illustration of the concept of ES6 class, prototypes, and function constructor

AIM:

ALGORITHM:

CODE:  


OUTPUT:



Q8) Illustration of CRUD in MongoDB using JavaScript

AIM:

ALGORITHM:

CODE:

CREATE.JS:

const { MongoClient } = require("mongodb");

const url = "mongodb://127.0.0.1";

const dbName = "interviews";

const client = new MongoClient(url);

try {

client.connect();

console.log("Successfully connected to MongoDb");

} catch (err) {

console.log("Error:", err);

}

const db = client.db(dbName);

async function main() {

let companiesCollection = db.collection("companies");

let newCompany = {

name: "Microsoft",

website: "www.microsoft.com",

location: "USA",

}

let result = await companiesCollection.insertOne(newCompany);

console.log("Inserted documents ID: ", result.insertedId);

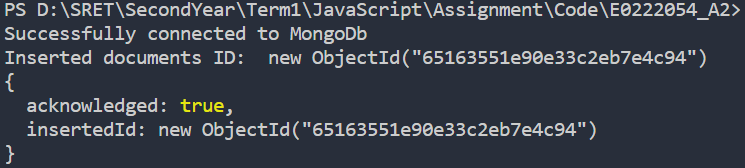
console.log(result);

client.close();

}

main();

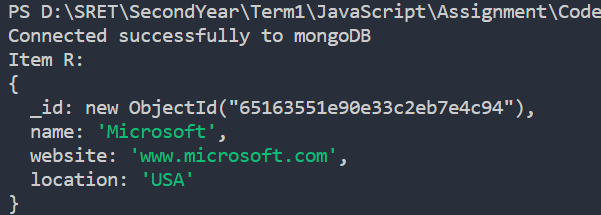
OUTPUT:



READ.JS:



OUTPUT:



UPDATE.JS:



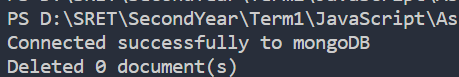
OUTPUT:



DROP.JS:



OUTPUT:



Q9) Design HTML page using pug Template Engine

AIM:

ALGORITHM:

CODE:

INDEX.JS

var express = require('express');

var app = express();

app.set('view engine', 'pug');

app.set('views', './views');

*// app.get('/',function(req,res){*

*//     res.render('dynamic',{*

*//         name: "TutorialsPoint",*

*//         url:"http://www.tutorialspoint.com"*

*//     });*

*// });*

app.get('/',function(req,res){

    res.render('application');

});

app.listen(3000,function(){

    console.log("Live at Port 3000");

});

SAMPLE.PUG

doctype html

html

    head

        title= "Hello Pug"

    body

        h1.classsname#idname "Hello World"

        h2 welcome to pug

        p this is a paragraph

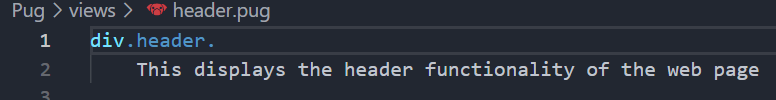
        p this is another paragraph

        example for multiline paragraph

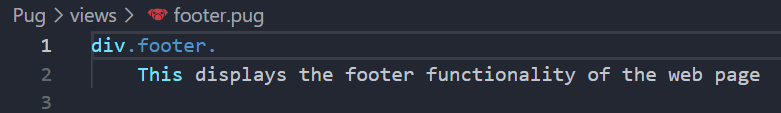
        div

            WHen more text to be displayed then pipe operwufbwiebfiuwnvwhfuwuwdajsccnkajscsnoaussffjjasddjjdhaosoaousdsouadhahnndkfj9udhq08wjqkwlwndiuuhch98hjejncijibc9ujcklqwj0iqihfjk chqciyqw.

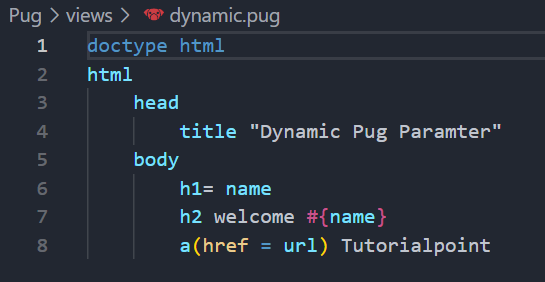
HEADER.PUG



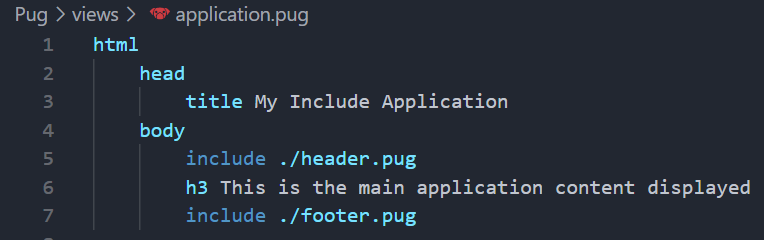
FOOTER.PUG



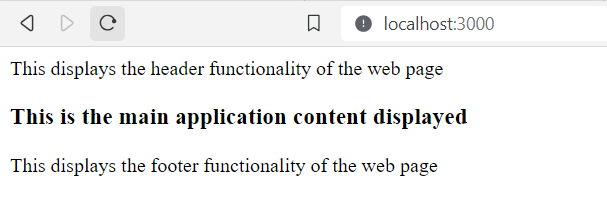
DYNAMIC.PUG



APPLICATION.PUG



OUTPUT:



Q10) Implement Search operation by Email ID from data stored in MongoDB using ExpressJS

AIM:

ALGORITHM:

CODE:

INDEX.JS

const express=require('express');

const mongoose=require('mongoose');

const bodyParser=require('body-parser');

const app=express();

app.use(bodyParser.urlencoded({extended:true}));

app.set("view engine","pug");

app.set('views','./views');

try

{

    mongoose.connect("mongodb://127.0.0.1/admission",{

        useNewUrlParser:true,

        useUnifiedTopology:true

    })

    console.log('Connected to mongoDB');

}

catch(err)

{

    console.log('Error Connection to mongoDB:',err);

}

const Schema=mongoose.Schema;

const DataSchema=new Schema({

    name:String,

    email:String,

});

const DataModel = mongoose.model("students",DataSchema);

app.get('/',(req,res)=>{

    res.render('form');

});

app.post('/submit',(req,res)=>{

    const { name, email }= req.body;

    const newData=new DataModel({ name, email });

    newData

        .save()

        .then(()=>{

            res.send('Student information saved to MongoDB!');})

        .catch((err)=>{

            console.log("Error savings data:",err);

            res.send("Error savings data.");

        });

});

app.get("/data", (req, res) => {

    const searchQuery = req.query.search || '';

    const searchFilter = { $regex: searchQuery, $options: "i" };

    DataModel.find({ email: searchFilter })

      .then(records => {

        res.render('data', { records, searchQuery });

      }).catch(\_ => res.send("Unexpected error"));

  })

  app.get('/update/:id', (req, res) => {

    const dataId = req.params.id;

    DataModel.findById(dataId).then(record => {

      res.render('update', { record })

    }).catch(\_ => res.send("unexpected error"))

  })

  app.post("/update/:id", (req, res) => {

    const recordId = req.params.id;

    const { name, email } = req.body;

    DataModel.findByIdAndUpdate(recordId, { name, email })

      .then(() => {

        console.log("Student Details updated");

        res.redirect("/data");

      })

      .catch((err) => {

        console.error("Error updating record:", err);

        res.send("Error updating record.");

      });

  });

  app.get("/delete/:id", (req, res) => {

    const recordId = req.params.id;

    DataModel.findByIdAndRemove(recordId)

      .then(() => {

        console.log("Student Record removed from MongoDB");

        res.redirect("/data");

      })

      .catch((err) => {

        console.error("Error removing record:", err);

        res.send("Error removing record.");

      });

  });

  app.listen(3000, () => {

    console.log('connected to portnumber 3000');

  });

UPDATE.PUG

doctype html

html

  head

    title Update Student Details

  body

    h1 Update Student Data

    form(action=`/update/${record.\_id}` method="POST")

      label(for="name") Name:

      input(type="text" name="name" value=record.name required)

      label(for="email") Email:

      input(type="email" name="email" value=record.email required)

      button(type="submit") Update

FORM.PUG

html

    head

        title Students details to MongoDB

    body

    h1 submit students details to mongodb

    form(action='/submit' method='POST')

        label(for='name') Name:

        input(type='text' name='name' required)

        label(for='email') Email:

        input(type='email' name='email' required)

        button(type="submit") submit

DATA.PUG

html

    head

        title Students details to MongoDB

    body

    h1 submit students details to mongodb

    form(action='/data' method='GET')

        label(for='search') Search by Email:

        input(type="Email" name="search")

        button(type="submit") Submit

    ul

        each record in records

            li

                strong Name:

                span=record.name

                br

                strong Email:

                span=record.email

                br

                a(href=`/update/${record.\_id}`) Edit

                span |

                a(href=`/delete/${record.\_id}`) Delete

OUTPUT:

