//AND

var AND\_array = [];

var store\_function = document.getElementById('function\_selection');

function ANDgate(AND\_line, time)

{

var AND\_currentLine;

var AND\_logicOutput;

var AND\_prevlogic;

var AND\_findLL;

/\*For loop below searches AND\_array for the line called, then stores the name to

use later.

\*/

for (var AND\_lineFinder = 0; AND\_lineFinder < AND\_array.length; AND\_lineFinder++)

{

if (AND\_line = AND\_array[AND\_lineFinder].name)

{

AND\_currentLine = AND\_array[AND\_lineFinder];

break;

}

}

/\*For loop finds the logic\_line the name belongs to and stores the logic state

the line is currently in.

\*/

for (var LL\_index = 0; LL\_index < logic\_lines.length; LL\_index++)

{

if (logic\_lines[LL\_index].name == AND\_line)

{

AND\_findLL = 'logic\_lines[' + LL\_index + ']';

AND\_prevlogic = eval(AND\_findLL).logic\_state;

break;

}

}

/\*For loop checks if each input line is HIGH> If not, the logic state of the gate is

set to LOW and the for lop is immediately exited.

\*/

for (var AND\_inputIter = 0; AND\_inputIter < AND\_currentLine.inputs.length; AND\_inputIter++)

{

AND\_logicOutput = 'HIGH';

if (eval(AND\_currentLine.inputs[AND\_inputIter]).logic\_state == "LOW")

{

AND\_logicOutput = 'LOW';

break;

}

}

/\*If statement checks if the logic state of the line changed. If so, the current time

is stored into the current line's inputsi

\*/

if (AND\_logicOutput != AND\_prevlogic)

{

eval(AND\_findLL).inputs.push(time);

document.getElementById('test').innerHTML = eval(AND\_findLL).inputs;

order\_existingArray(LL\_index);

document.getElementById('demo1').innerHTML =

document.getElementById('demo1').innerHTML +

'<br>' +

eval(AND\_findLL).inputs[eval(AND\_findLL).input\_iter] +

', was ' +

eval(AND\_findLL).logic\_state;

/\*if(AND\_logicOutput == "HIGH")

{

eval(AND\_findLL).logic\_state = AND\_logicOutput;

}

else

{

eval(AND\_findLL).logic\_state = AND\_logicOutput;

}\*/

}

//IC line names have to be eval()'d from AND\_array eval(AND\_array[0].inputs[0])

/\*if (logic\_lines[0].logic\_state == 'HIGH')

{

document.getElementById('test').innerHTML = "Test text";

}\*/

}

function ANDgate\_setup()

{

//document.getElementById('demo').innerHTML = logic\_lines[0].name;

//Setup needs to create a block to enter parameters

var table\_inputs = "<table><tr><th>AND</th></tr>";

for (var table\_count = 0; table\_count < logic\_lines.length; table\_count++)

{

table\_inputs += "<tr><td class='inputTable'>" +

logic\_lines[table\_count].name +

"</tr></td>";

}

table\_inputs += "</table>" +

"<button id='createObject'>Finish Gate</button>";

document.getElementById('moreLogic').innerHTML = table\_inputs;

$(document).ready(function()

{

$(".inputTable").click(function()

{

$(this).toggleClass("selected\_input");

});

$("#createObject").click(function()

{

// IC\_nameLF grabs name from name textbox

var IC\_nameLF = document.getElementById('create\_name').value;

//Gets the amount of names with 'selected\_input' class

var AND\_classCounter = document.getElementsByClassName('selected\_input');

//Used to store gate name and input location for gate

var AND\_inputObject = {name: IC\_nameLF, inputs:[]};

/\*Two for loops are used to get the name for each table element with the selected\_input class

and finds which logic line the name belongs to.

\*/

for (var iter\_classCounter = 0; iter\_classCounter < AND\_classCounter.length; iter\_classCounter++)

{

for (var LL\_index = 0; LL\_index < logic\_lines.length; LL\_index++)

{

/\*When the line is found, the if statement stores the name into the inputs and breaks for loop

for (hopefully) efficient processing.

\*/

if (logic\_lines[LL\_index].name == AND\_classCounter[iter\_classCounter].innerHTML)

{

AND\_inputObject.inputs.push('logic\_lines['+ LL\_index +']');

break;

}

}

}

AND\_array.push(AND\_inputObject);

store\_function.value = 'ANDgate("' + IC\_nameLF + '", logic\_timing)';

document.getElementById('moreLogic').innerHTML = "";

});

});

//To allow the IC Creator to access setup info: update function flag, gate function name

//Hold values to that multiple lines can use the same gate without interfering with eachother

//Needs the main gate function to access line information

}