

INTRODUCTION TO PROGRAMMING

HTML, CSS and JavaScript program to check if a number is prime or not.



By Kayleigh Lamb
(15th Dec 2012)

Contents

Task & Program Requirements	Page 2
Version 1 Overall Flowchart – Feature 1	Page 3
Version 1 Flowchart – parse_number() – Feature 1	Page 4
Version 1 Flowchart – is_valid()	Page 5
Version 1 Flowchart – is_prime()	Page 6
Version 1 Code – Feature 1 Program only	Pages 7 - 8
Version 1 Testing Schedule – Feature 1 Program only	Page 9
Version 1 Overall Flowchart – Feature 2	Page 10
Version 1 Code – Feature 2 Program only	Pages 11 - 12
Version 1 – Testing Schedule – Feature 2 Program only	Page 13
Version 2 – Optimisations & Bug fixes	Page 14
Version 2 Overall Flowchart	Page 15
Version 2 Flowchart – valid()	Page 16
Version 2 Flowchart – is_prime()	Page 17
Version 2 Flowchart – feature1()	Page 18
Version 2 Flowchart – feature2()	Page 19
Version 2 Code	Pages 20 - 23
Version 2 –Testing Schedule – Feature 1	Page 24
Version 2 –Testing Schedule – Feature 2	Page 25
Version 3 – Optimisations & Bug fixes	Page 26
Version 3 Overall Flowchart	Page 27
Version 3 Flowchart – valid()	Page 28
Version 3 Flowchart – is_prime()	Page 29
Version 3 Flowchart – feature1()	Page 31
Version 3 Flowchart – feature2()	Page 32
Version 3 Code	Pages 33 - 37
Version 3 –Testing Schedule – Feature 1	Page 38
Version 3 –Testing Schedule – Feature 2	Page 39
Summary Report	Page 40

Introduction to Programming

Task

You have just started a new job with a small company that produces spreadsheet tools. Your boss has an idea for two new features to be added to the latest web based spreadsheet tool. The features are:

Requirements

Feature 1 - The program must analyse a number entered by the user, and tell them if the number is a prime or not.

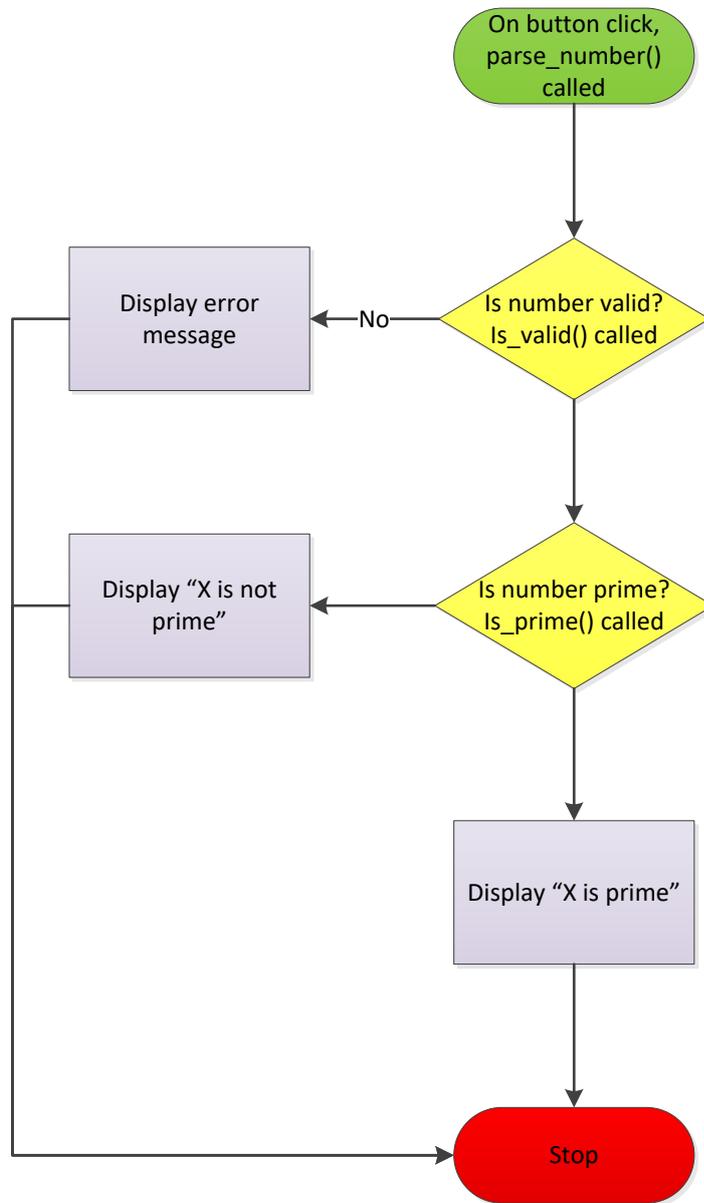
Feature 2 - The program must also be able to generate a list of prime numbers between an upper and a lower integer number which are also entered by the user.

1. The two features are incorporated in to a single program; the user needs to be able to select which of the two features are required.
2. For feature 1, the user needs to enter a single number; the number is only valid if it is a round integer and greater than 1.
3. For feature 2, the user needs to enter both lower and upper bound numbers. The numbers entered are only valid if they are round integer numbers and also greater than 1. The lower number must also have a lower value than the upper number.
4. Number 1 is not considered prime.
5. Number 2 is considered prime.
6. All even numbers are not prime as they are divisible exactly by 2, so 4, 6, 8 etc are not prime.
7. For feature 1, once the tool has decided whether or not a number is a prime, a message needs to be displayed in the following format to the user "X is prime" or "X is not prime".
8. For feature 2, a list of the relevant prime numbers (if any), need to be displayed to the user in the form "X is prime".
9. The program should run as quickly as possible.

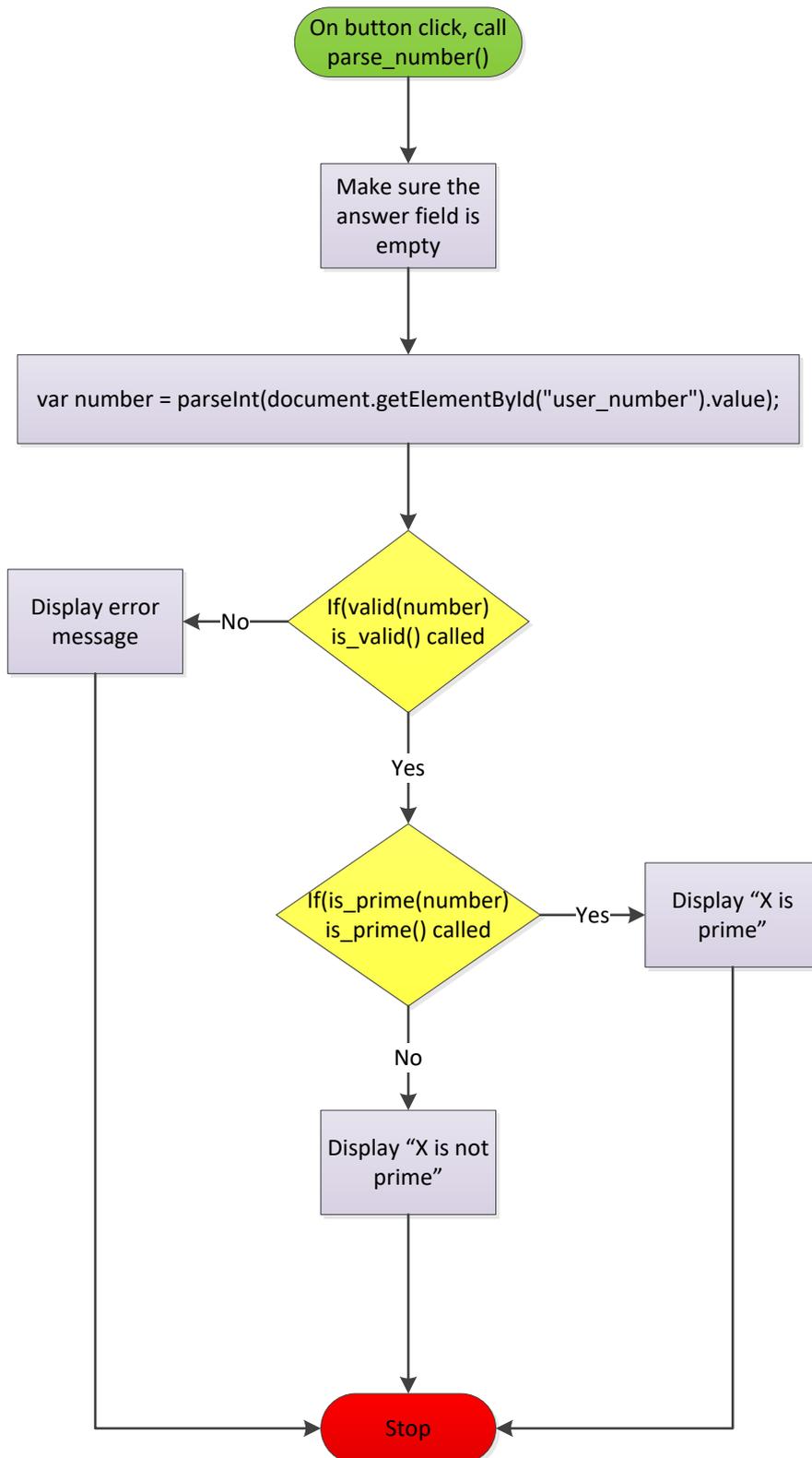
The program should also meet the following general requirements for a good piece of software:

1. Appropriate variable names
2. Good use of comments
3. Simple and logical design
4. Software should be fully tested
5. Code can make good use of JavaScript functions
6. The program should be a result of your own design

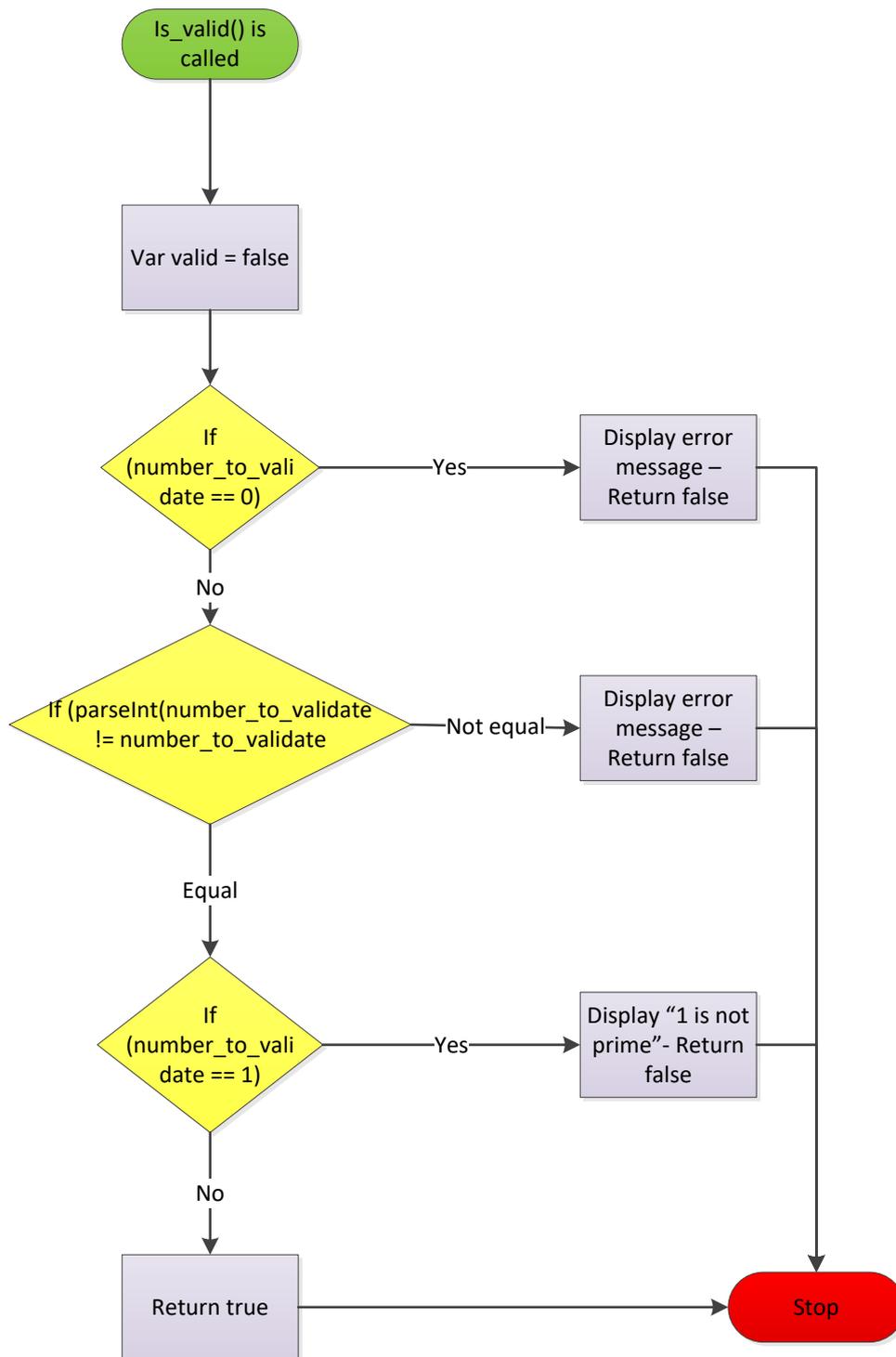
Overall Flowchart V1 – Feature 1 Only



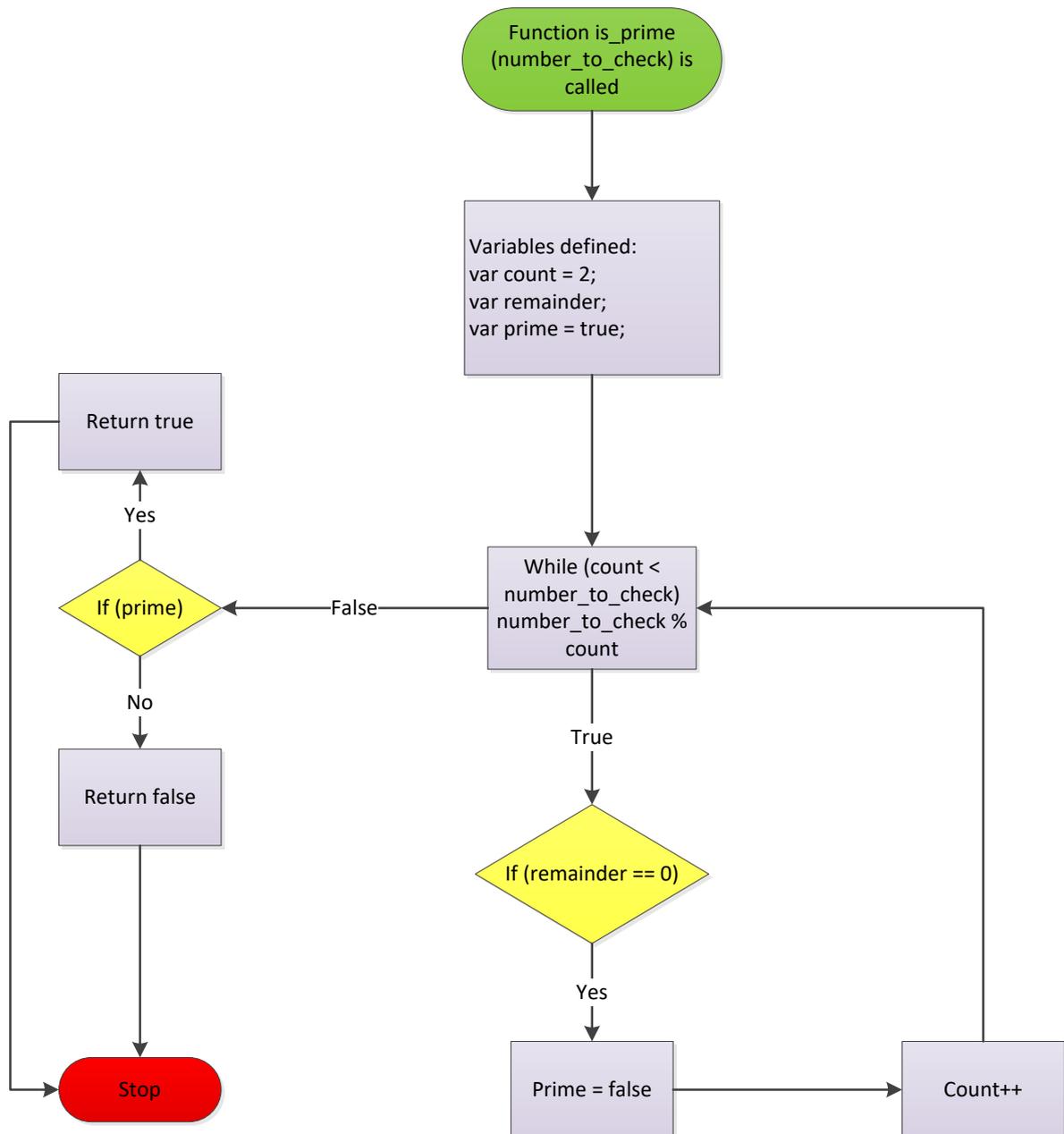
Parse_number – V1 Flowchart



Is_valid() V1 Flowchart



Is_prime() – Version 1 Flowchart



Version 1 Code - Feature 1 Only

```

<html>
  <head>
    <script type="text/javascript">

      function parse_number(){
        //Make sure the answer field is empty
        answer1.innerHTML = '';
        //Number entered by user
        var number = parseInt(document.getElementById("user_number").value);
        //Call the valid function to check if ok to test if prime
        if (is_valid(number)){
          //Call the prime function and display if prime or not
          if (is_prime(number)){
            answer1.innerHTML = "<font color=red>" + number + "</font>" + "
is prime, ";
          }
          else{
            answer1.innerHTML = "<font color=red>" + number + "</font>" + "
is NOT prime, ";
          }
        }
        //Function to check number entered is valid
        function is_valid(number_to_validate){
          //Number is not classed as valid until proven otherwise
          var is_valid = false;

          //Number entered must be greater than 1
          if(number_to_validate == 0){
            answer1.innerHTML = "Please enter a number greater than 1, ";
            return false;
          }

          //Number entered must be a whole integer
          if(parseInt(number_to_validate) != number_to_validate){
            answer1.innerHTML = "Please enter a whole number, ";
            return false;
          }

          //Number 1 is not prime
          if(parseInt(number_to_validate) == 1){
            answer1.innerHTML = "<font color=red>" + number_to_validate +
"</font>" + " is NOT prime, ";
            return false;
          }
          return true;
        }

        //Function to check if the number entered is prime - using true or false
        boolean
        function is_prime(number_to_check){

          var count = 2; //Start by testing the number 2
          var remainder; //Checks for a remainder after modulus test
          var prime = true; // Start by defining all numbers as prime

          //While count is less than the number_to_check, test for a remainder
          using modulus
          while(count < number_to_check){
            remainder = number_to_check % count;
            //If there is a remainder, prime = false
            if(remainder == 0){
              prime=false;
            }
          }
        }
      }
    </script>
  </head>
</html>

```

```

        break;
    }
    //Add 1 to count each time the program goes through the loop
    count++;
}
//If prime, return true, else return false
if (prime){
    return true;
}
else{
    return false;
}
}
}
</script>

<!-- begin HTML input box-->
<body>
    <table>
        <tr>
            <td>
                <h1><font color="red"><em>Prime Number Calculator
1</em></font></h1>
                <br>
                <h4>Use this tool to calculate if a single number is prime.</h4>
                <form>
                    Enter number: <input type="text" id="user_number" value="0">
                    <!-- Calculate button - call the function on click-->
                    <input type="button" value="Calculate"
onclick="parse_number()"></input>
                    <!-- Reload page button-->
                    <input type="button" value="Reset Program"
onClick="document.location.reload(true)">
                </form>
            </td>
        </tr>
    </table>
    <div id='answer1'>
    </div>
</body>
</html>

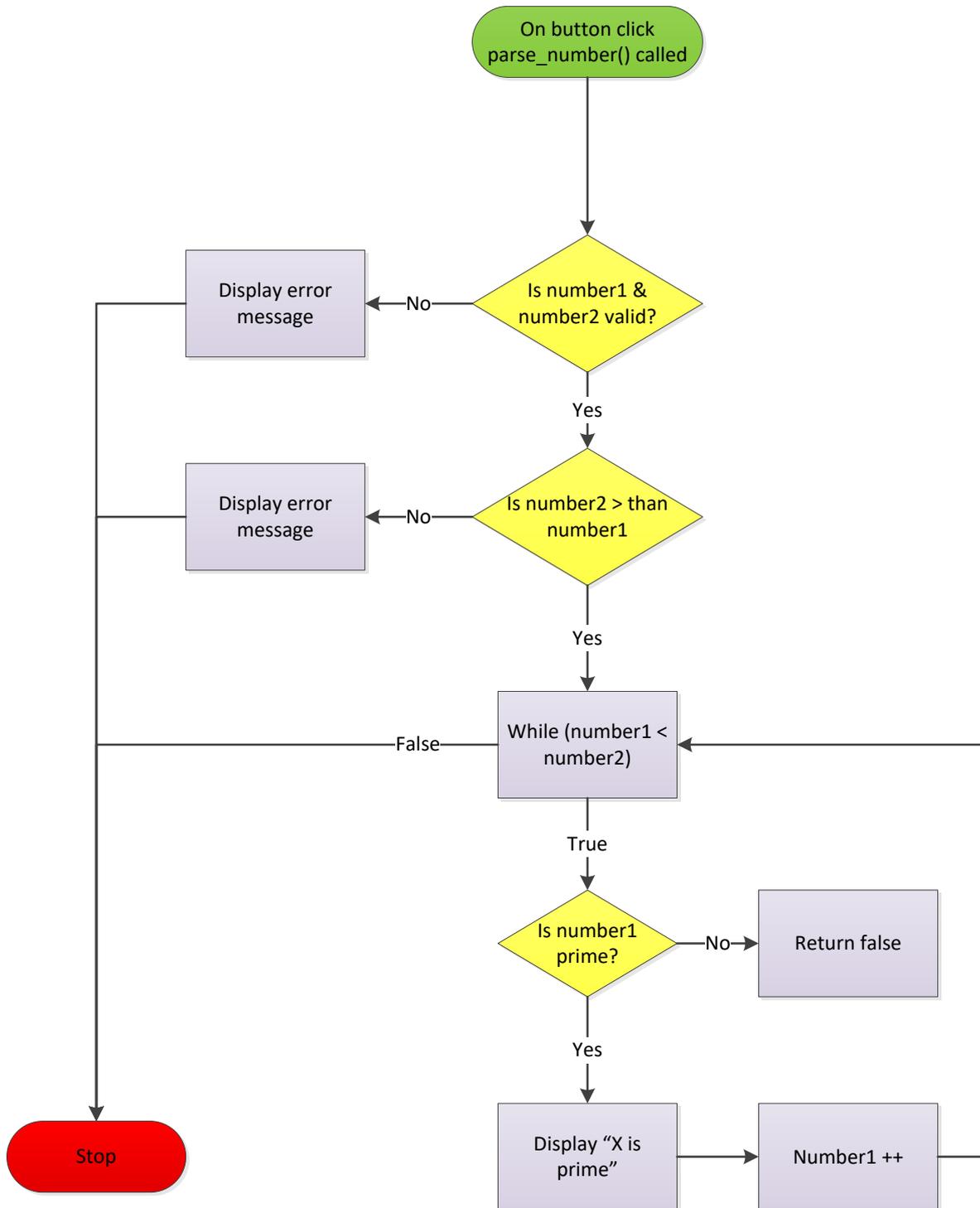
```

Testing Schedule for Version 1 - Feature 1 only

NOTE: All tests performed in Google Chrome version 23.0.1271.64 m, Firefox version 17.0.10 & Internet Explorer version 9.0.10 (ALL browsers were up to date at time of test 24/11/2012)

Test No.	Description	Expected Outcome	Actual Outcome	Test by	Chrome Pass?	IE Pass?	Firefox Pass?	Action?
1 Check if prime in feature 1								
1a	Test the number 1	"1 is NOT prime"	"1 is NOT prime"	Kayleigh	✓	✓	✓	None
1b	Test the number 2	"2 is prime"	"2 is prime"	Kayleigh	✓	✓	✓	None
1c	Test the number 23	"23 is prime"	"23 is prime"	Kayleigh	✓	✓	✓	None
1d	Test the number 99	"99 is NOT prime"	"99 is NOT prime"	Kayleigh	✓	✓	✓	None
1e	Test the number 99,999	"99,999 is NOT prime"	"99,999 is NOT prime"	Kayleigh	✓	✓	✓	None
1f	Test the number 9007199254740992	"9007199254740992 is NOT prime"	"101 is prime"	Kayleigh	✓	✓	✓	None
2 Number must be a whole integer in feature 1								
2a	Test the number 0.2	"Please enter a number greater than 1"	"Please enter a number greater than 1"	Kayleigh	✓	✓	✓	None
2b	Test the number 2.9	"Please enter a whole number"	"2 is prime"	Kayleigh	✗	✗	✗	Adjust the validate function - issue with the number being parsed before it is validated
2c	Test the number 100.6	"Please enter a whole number"	"100 is NOT prime"	Kayleigh	✗	✗	✗	Adjust the validate function - issue with the number being parsed before it is validated
2d	Test the word "rhubarb"	"Please enter a whole number"	"Please enter a whole number"	Kayleigh	✓	✓	✓	None
2e	Test the number 3478.56	"Please enter a whole number"	"3478 is NOT prime"	Kayleigh	✗	✗	✗	Adjust the validate function - issue with the number being parsed before it is validated
2f	Test the number 2,300,456,9876	"Please enter a whole number"	"2300456" is NOT prime"	Kayleigh	✗	✗	✗	Adjust the validate function - issue with the number being parsed before it is validated

Overall Flowchart V1 Feature 2



Version 1 Code - Feature 2 Only

```

<html>
  <head>
    <script type="text/javascript">
      //Function to parse number and print to the screen
      function parse_number(){
        //make sure the answer field is empty
        answer.innerHTML = '';

        //1st number entered by user
        var number1 =
parseInt(document.getElementById("user_number1").value);
        //2nd number entered by user
        var number2 =
parseInt(document.getElementById("user_number2").value);

        //Calls the valid function to check number1 & number2
        if(is_valid(number1) && is_valid(number2)){
          //Number2 is greater than number1, enter the while loop else
display error message
          if(number2 > number1){
            while(number1 < number2){
              //While number1 is less than number2, test if it's
prime by calling the prime function
              if(is_prime(number1)){
                //If prime, display it on the screen
                answer.innerHTML += "<font color=red>" + number1
+ "</font>" + " is prime, <br>";
              }
              //Add 1 to number1 each time the program goes through
the loop
              number1 ++;
            }
          }
          else{
            answer.innerHTML = "the second number needs to be higher
than the first one!";
          }
        }

        //Check if number is valid
        function is_valid(number_to_validate){
          //False until proven otherwise
          var is_valid = false;

          //number entered must be greater than 1, if not display error
message
          if (number_to_validate == 0){
            answer.innerHTML = "Please ensure both numbers are greater
than 1, ";
            return false;
          }

          //Number entered must be a whole integer, if not display an error
message
          if (parseInt(number_to_validate) != number_to_validate){

```

```

        answer.innerHTML = "Please ensure both numbers are a whole
number, ";
        return false;
    }
    //Else return true
    return true;
}
//Check if number is prime function
function is_prime(number_to_check){
    var count = 2; //Start by testing against 2
    var remainder; //Test for a remainder using modulus
    var prime = true; //Start by defining all as prime

    while(count < number_to_check){
        //While count is less than the number to check, test for
remainder
        remainder = number_to_check % count;
        if(remainder == 0){
            //If remainder = 0, then prime is false
            prime=false;
            break;
        }
        //Add 1 to count each time the program goes through the
loop
        count++;
    }
    //if prime, return true, else return false
    if (prime){
        return true;
    }
    else{
        return false;
    }
}

</script>
</head>
<body>
    <table>
        <tr>
            <td>
                <h1><font color="red"><em>Prime Number Calculator
2</em></font></h1>
                <br>
                <h3>Determines the prime numbers between two numbers
entered</h3>
                <form>
                    Enter number 1: <input type="text" id="user_number1"
value="0">
                    Enter number 2: <input type="text" id="user_number2"
value="0">
                    <input type="Button" value="Calculate"
onclick="parse_number()" onsubmit="return false"></input>
                </form>
            </td>
        </tr>
    </table>
    <div id='answer'>
</div>
</body>
</html>

```

Testing Schedule for V 1.0 - Feature 2

NOTE: All tests performed in Google Chrome version 23.0.1271.64 m , Firefox version 17.0 & Internet Explorer version 9.0.10 (ALL browsers were up to date at time of test 24/11/2012)

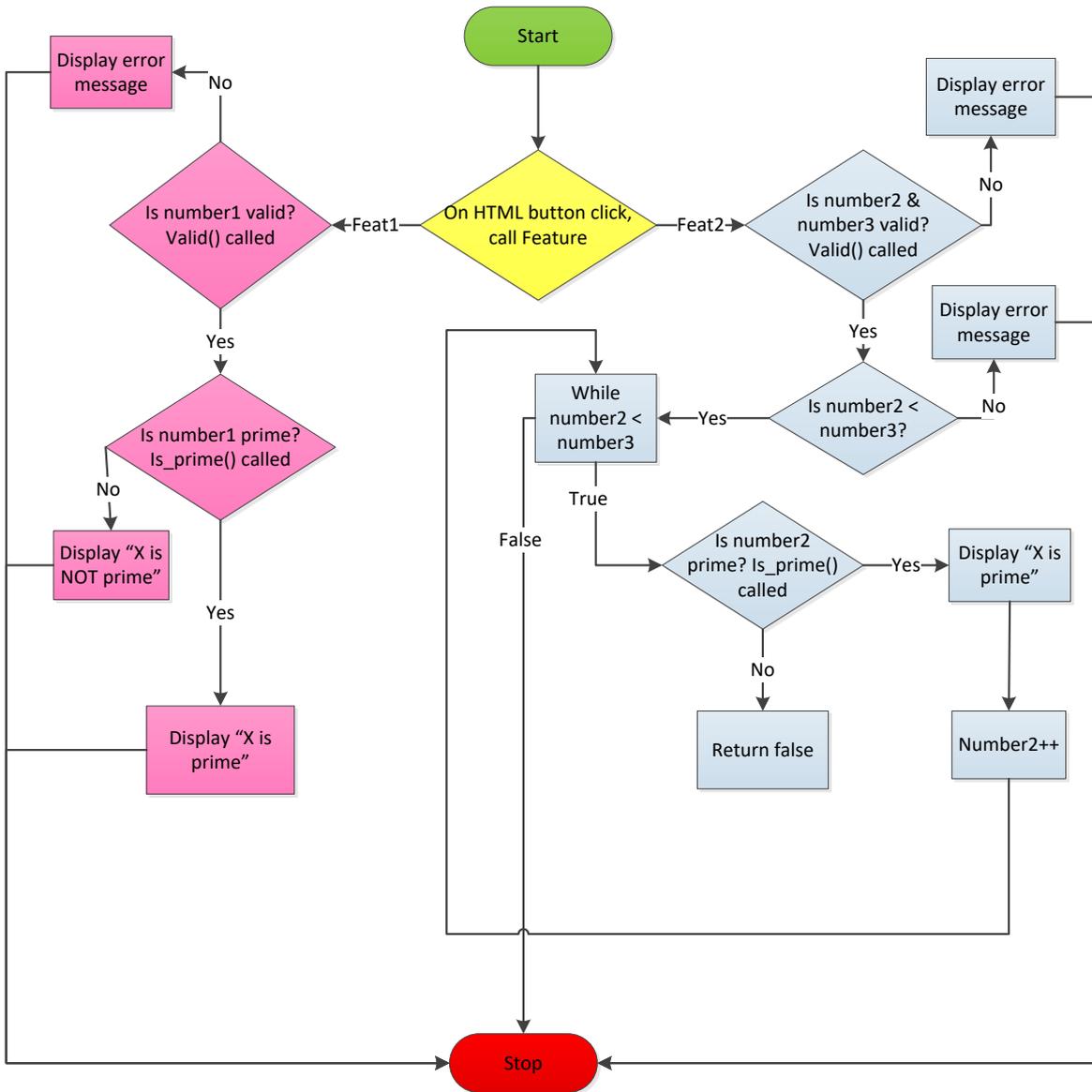
**Reference list used to check myprimes are correct can be found here: - <http://primes.utm.edu/lists/small/1000.txt>

Test No.	Description	Expected Outcome	Actual Outcome	Test by	Chrome Pass?	IE Pass?	Firefox Pass?	Action?
3 ** REQUIREMENTS - Check for a range of primes in Feature 2								
3a	Test the number range 1 to 100	All primes in range print to screen	Prints out all primes, but thinks 1 is prime	KL	✗	✗	✗	Need to add an if statement to remove 1 as prime
3b	Test the number range 5 to 600	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	None
3c	Test the number range 23 to 9,999	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	Worked, but took 20 secs - will try to make more
3d	Test the number range 300 to 7,908	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	Worked and took 10 seconds
3e	Test the number range 89 to 104	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	None
3f	Test the number range 1,000 to 2,000	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	None
3g	Test the number range 2389 to 67,000	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	Worked, but took 3 minutes in firefox - will try to make more efficient
4 REQUIREMENTS - Number must be a whole integer in feature 2								
4a	Input 1 fraction: 0.6 & 100	"Please ensure both numbers are greater than 1"	"Please ensure both numbers are greater than 1"	KL	✓	✓	✓	None
4b	Input 1 fraction box 2: 26 & 500.78	"Number entered must be a whole integer."	All primes in range print to screen	KL	✗	✗	✗	Incorporate bug fix to stop user entering a fraction in the 2nd box
4c	Input one fraction > 1 in box 1: 34.87 & 320	"Number entered must be a whole integer."	All primes in range print to screen	KL	✗	✗	✗	Incorporate bug fix to stop user entering a fraction in the 2nd box
4d	Input 2 fractions: 89.5 & 976.45	"Number entered must be a whole integer."	All primes in range print to screen	KL	✗	✗	✗	Incorporate bug fix to stop user entering a fraction in the 2nd box
5 REQUIREMENTS - Lower bound number must actually be lower than the higher bound number								
5a	Test numbers 100 & 3	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None
5b	Test numbers 1000 & 378	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None
5c	Test numbers 10 & 1	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None
5d	Test numbers 16,000 & 650	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None

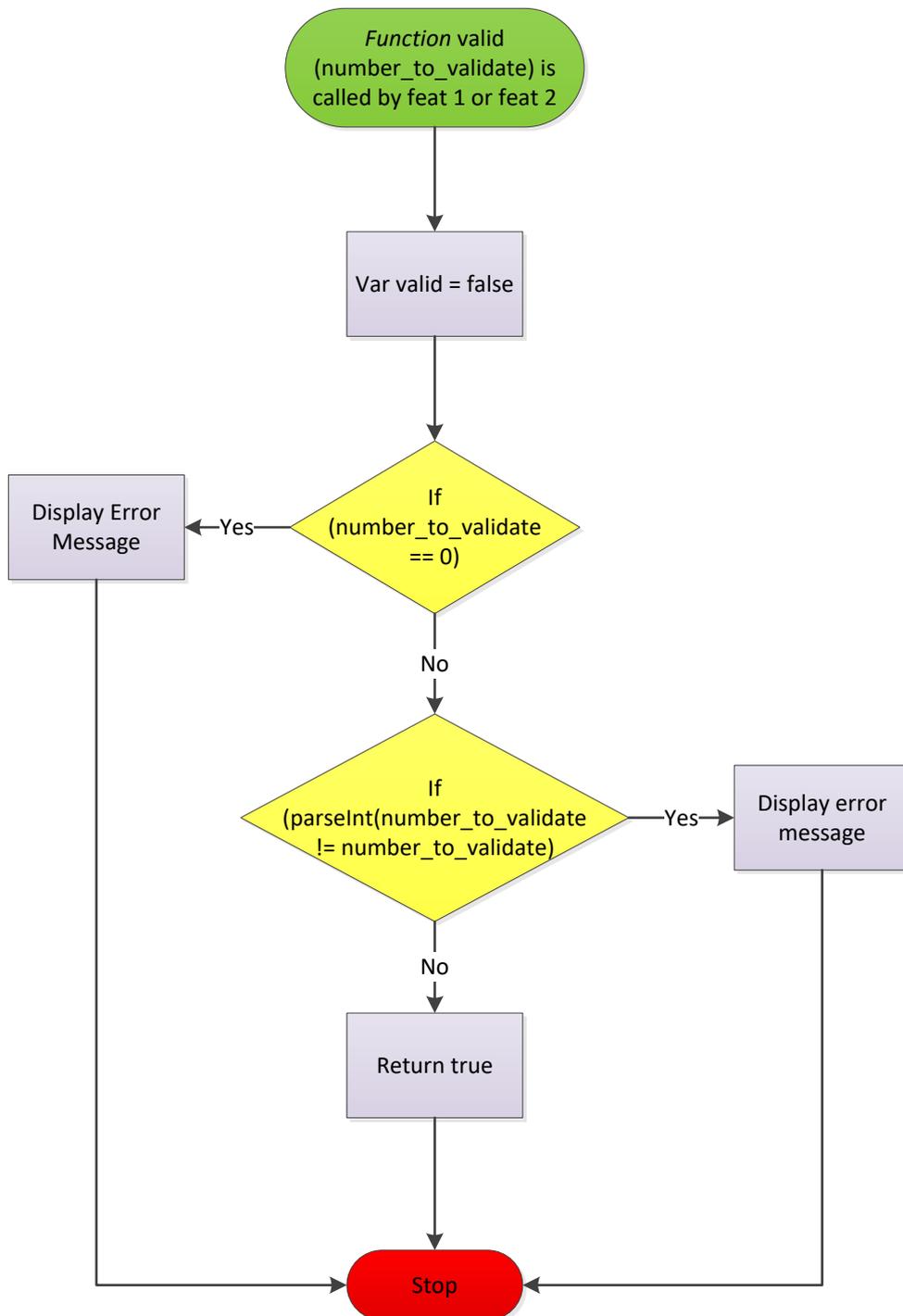
Optimisations & Bug fixes incorporated for V 2.0

No.	Bug/Description	Fix
1	Feature 1 & 2 incorporated into the same program	As per the requirements
Optimisations		Result
2	Made use of the same prime checker function for both feature 1 & 2	More efficient than having the same function twice in 2 different programs.
3	Introduced testing up to the square root of a number only, in the prime checker function	A massive increase when using feature 2, ranges returned much quicker - As per testing
4	Declared all functions before they are called	More efficient as the browser deciphers them before they are called
5	Created a more friendly looking user interface	Looks better
6	Introduced CSS for font header styling	More efficient than using inline styles

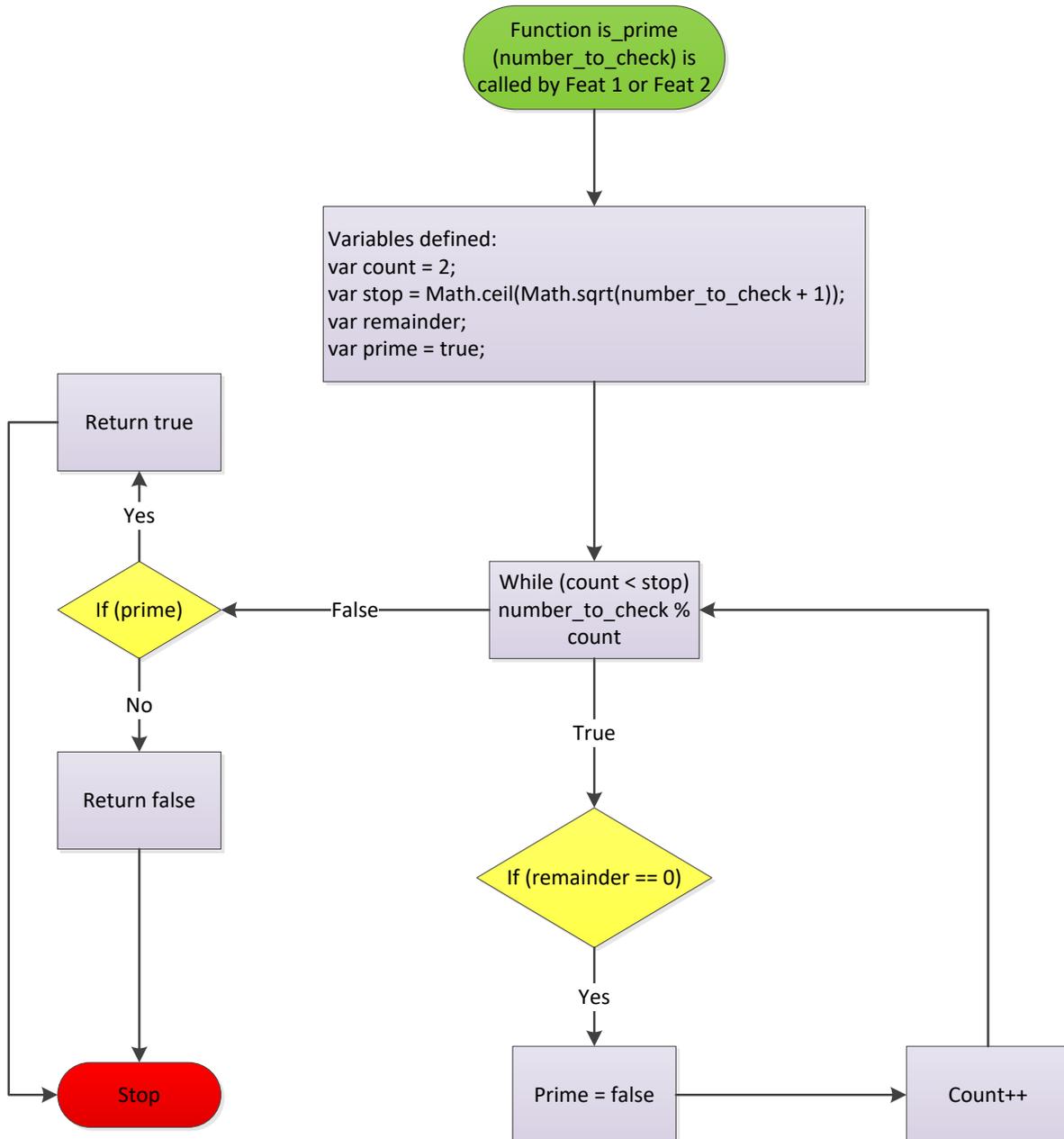
Overall Flowchart for V2



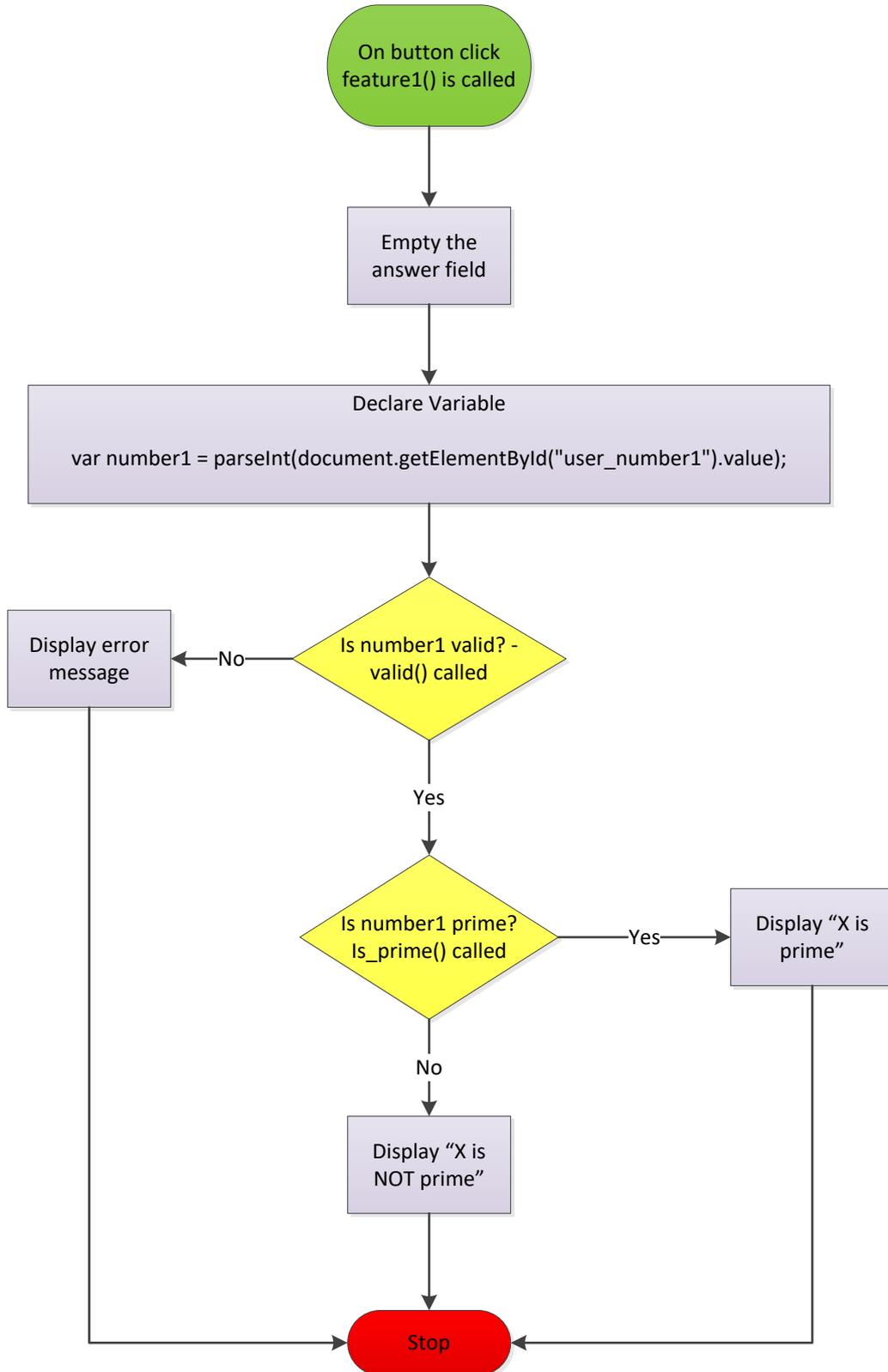
Function valid (number_to_validate) V2



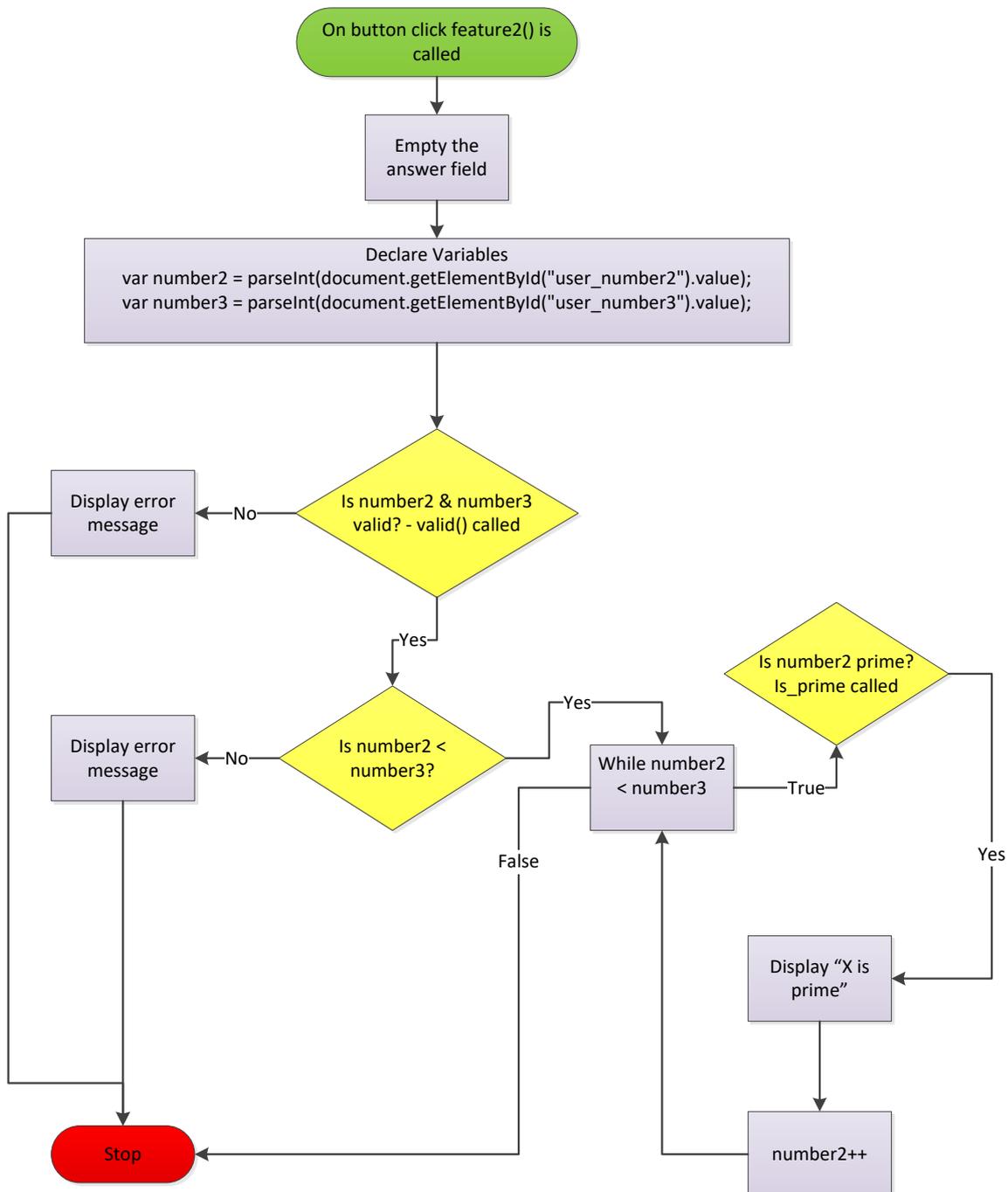
Is_prime() – Version 2 Flowchart



Function feature1() V2 Flowchart



Function feature2() V2 Flowchart



Version 2 Code - Feature 1 & 2 Concatenated

```

<html>
  <head>
    <style type="text/css">
      /* CSS Styling for page - including font type, size & colours*/

      h1 {
        color:#CC0099;
        font-family: sans-serif;
        font-style:italic;
      }
      h2 {
        color:#CC0099;
        font-family: sans-serif;
      }
      h4 {
        color:#CC0099;
        font-family: sans-serif;
      }
      p {
        font-family: sans-serif;
        font-size: 15px;
      }
      div{
        color:#CC0099;
        font-family: sans-serif;
        font-size: 15px;
      }
      body{
        background-color: #99CCFF;
      }
      input {
        font-family: sans-serif;
      }
    </style>
    <script type="text/javascript">

      //Validate function
      function valid(number_to_validate){

        //Guilty until proven otherwise
        var valid = false; //guilty until proven otherwise

        //Number entered must be greater than 1
        if(number_to_validate == 0) {
          answer1.innerHTML = "Please ensure any numbers entered
are greater than 1, ";
          return false;
        }
        //Number entered must be a whole integer, if not, return
false
        /**NEEDS WORK AS FUNCTION DOESN'T WORK
        if(parseInt(number_to_validate) != number_to_validate) {
          answer1.innerHTML = "Please ensure numbers entered are
a whole number, ";
          return false;
        }
        //Else return true

```

```

        return true;
    }

    //check if prime function
    function is_prime(number_to_check){
        var count = 2; //Start by testing the number against 2
        var stop = Math.ceil(Math.sqrt(number_to_check + 1));
//Stop when count has reached the sqrt of number rounded up + 1
        var remainder; //Modulus test for remainder
        var prime = true; //Start of by saying all numbers are
prime
        while(count < stop){
            //while count is less than stop, continue to test for a
remainder each time with the modulus operator
            remainder = number_to_check % count;
            if(remainder == 0){
                //If there is a remainder then prime = false
                prime=false;
                break;
            }
            //add one to count each time the program goes through
the loop
            count++;
        }
        if (prime){
            return true;
        }
        else{
            return false;
        }
    }

    //feature 1 program begin
    function feature1(){
        //make sure the answer field is empty
        answer1.innerHTML = '';

        //Number entered into feature 1 form
        var number1 =
parseInt(document.getElementById("user_number1").value);
        //Calls the valid function to check the number entered is
valid
        if(valid(number1) )
        {
            //Checks if the number is prime by calling the prime
function
            if (is_prime(number1)){
                //Displays "X is prime" if prime function returns
true
                answer1.innerHTML = + number1 + " is prime, ";
            }
            //Displays "X is NOT prime" is prime function returns
false
            else{
                answer1.innerHTML = + number1 + " is NOT prime, ";
            }
        }
    }

    //Feature 2 prgram begin

```

```

function feature2(){
    //Make sure the answer field is empty
    answer1.innerHTML = '';

    //Lower bound number entered into feature 2 form
    var number2 =
parseInt(document.getElementById("user_number2").value);
    //Upper bound number entered into feature 2 form
    var number3 =
parseInt(document.getElementById("user_number3").value);

    //Calling the valid function to check numbers are ok to be
tested
    if(valid(number2) && valid(number3) ){
        //Tests if number2 is less than number 3
        if(number2 < number3){
            while(number2 < number3){
                //Calls the is_prime function
                if(is_prime(number2)){
                    //Displays "X is prime" if prime function
returns true
                    answer1.innerHTML += + number2 + " is
prime, ";
                }
                //Continues to print out prime numbers up to
the higher bound number entered
                number2 ++;
            }
        }
        //Displays error message if number2 is higher than
number3
    }else{
        answer1.innerHTML = "The 2nd number must be higher
than the 1st!";
    }
}

</script>
</head>
<!-- Begin HTML input box-->
<body>
    <table border="6" align="center" valign="middle"
bordercolor="#CC0099" bgcolor="#ffffcc" width="95%" height="95"
cellspacing="10" cellpadding="20">
        <tr>
            <td align="center" height="10%" colspan="2">
                <h1>JavaScript prime number checker</h1>
            </td>
        </tr>
        <tr>
            <td align="left" width="50%" height="20%">
                <h2>Prime Number Calculator 1</h2>
                <br>
                <p>Use this tool to calculate if a single number is
prime.</p>
                <form><p>
                    Enter number: <input type="text" id="user_number1"
value="0" style="width: 100px;">
                    <!-- Calculate button - call the function on click-->

```

```

        <input type="button" value="Calculate"
onclick="feature1()"></input>
        </p>
        </form>
    </td>
    <td align="left" height="20%">
        <h2>Prime Number Calculator 2</h2>
        <br>
        <p>Use this tool to find a range of primes between two
given numbers.</p>
        <form><p>
            Enter number 1: <input type="text" id="user_number2"
value="0" style="width: 100px;">
            <br>
            Enter number 2: <input type="text" id="user_number3"
value="0" style="width: 100px;">
            <!-- Calculate button - call the function on click-->
            <input type="button" value="Calculate"
onclick="feature2()"></input>
            </p>
        </form>
    </td>
</tr>
<tr>
<td colspan="2" align="center" valign="top">
<h2> ANSWER </h2>
<div id='answer1'>
</div>
</td>
</tr>
<tr>
<td align="center" height="5%" colspan="2">
<p>Introduction to programming - Assignment by Kayleigh Lamb
</p>
</td>
</tr>
</table>
</body>
</html>

```

Testing Schedule for V.2.0 - Feature 1

NOTE: All tests performed in Google Chrome version 23.0.1271.64 m , Firefox version 17.0.0 & Internet Explorer version 9.0.10 (ALL browsers were up to date at time of test 24/1/2012)

Test No.	Description	Expected Outcome	Actual Outcome	Test by	Chrome Pass?	IE Pass?	Firefox Pass?	Action?
6 Check if prime in feature 1								
6a	Test the number 1	"1 is NOT prime"	"1 is prime"	KL	✗	✗	✗	Need to add an if statement to remove 1 as
6b	Test the number 2	"2 is prime"	"2 is prime"	KL	✓	✓	✓	None
6c	Test the number 23	"23 is prime"	"23 is prime"	KL	✓	✓	✓	None
6d	Test the number 99	"99 is NOT prime"	"99 is NOT prime"	KL	✓	✓	✓	None
6e	Test the number 99,999	"99,999 is NOT prime"	"99,999 is NOT prime"	KL	✓	✓	✓	None
6f	Test the number 9007199254740992	"9007199254740992 is NOT prime"	"9007199254740992 is NOT prime"	KL	✓	✓	✓	None
7 Number must be a whole integer in feature 1								
7a	Test the number 0.2	"Please ensure any nos entered are greater than 1"	"Please ensure any nos entered are greater than 1"	KL	✓	✓	✓	None
7b	Test the number 2.9	"Number entered must be a whole integer."	"2 is prime"	KL	✗	✗	✗	Still need to fix the parse number issue to
7c	Test the number 100.6	"Number entered must be a whole integer."	"100 is NOT prime"	KL	✗	✗	✗	Still need to fix the parse number issue to stop fractions
7d	Test the word "hubarb"	"Please ensure numbers entered are a whole no."	"Please ensure numbers entered are a whole no."	KL	✓	✓	✓	None
7e	Test the number 3478.56	"Number entered must be a whole integer."	"3478.56 is NOT prime"	KL	✗	✗	✗	Still need to fix the parse number issue to stop fractions
7f	Test the number 2,300,456,9876	"Number entered must be a whole integer."	"2300456 9876 is NOT prime"	KL	✗	✗	✗	Still need to fix the parse number issue to stop fractions

Testing Schedule for V 2.0 - Feature 2

NOTE: All tests performed in Google Chrome version 23.0.1271.64 m , Firefox version 17.0 & Internet Explorer version 9.0.10 (ALL browsers were up to date at time of test 24/11/2012)

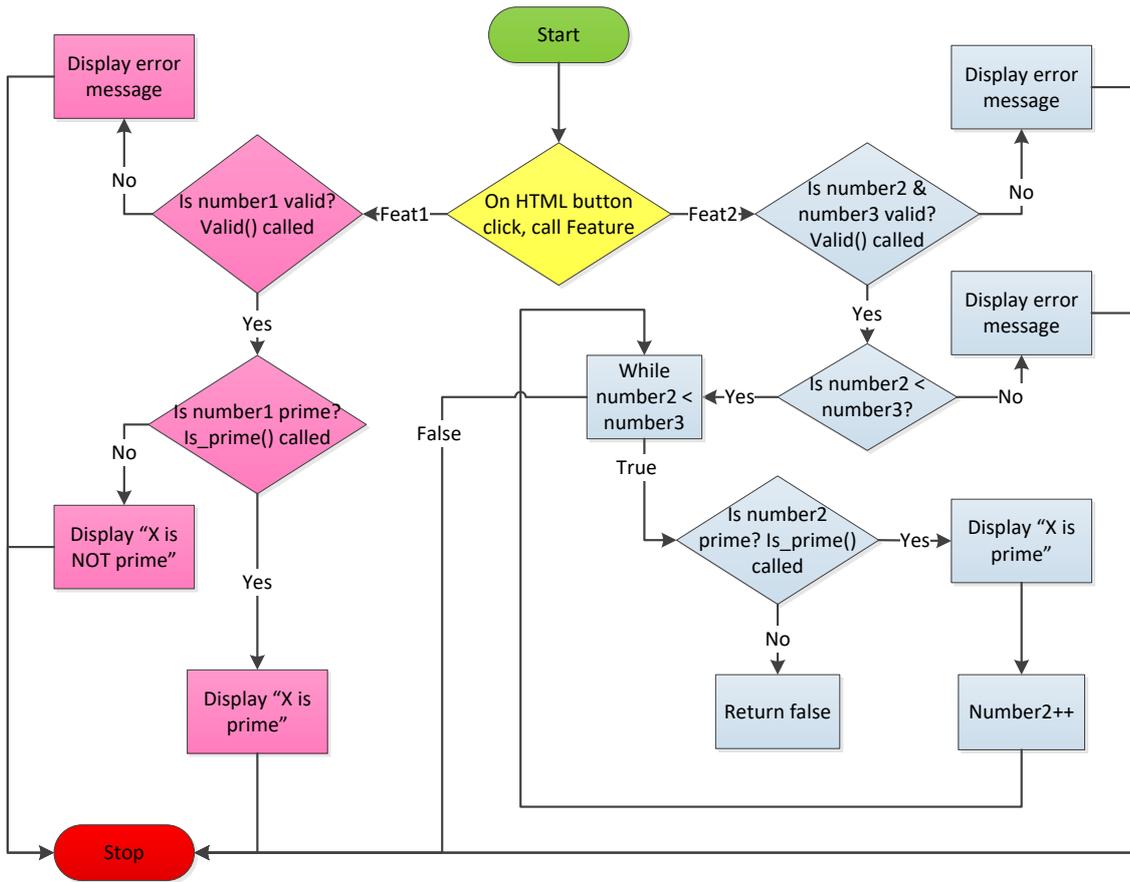
**Reference list used to check my primes are correct can be found here:- <http://primes.utm.edu/lists/small/1000.txt>

Test No.	Description	Expected Outcome	Actual Outcome	Test by	Chrome Pass?	IE Pass?	Firefox Pass?	Action?
3 ** REQUIREMENTS - Check for a range of primes in Feature 2								
3a	Test the number range 1 to 100	All primes in range print to screen	Prints out all primes, but thinks 1 is prime	KL	✗	✗	✗	Need to add an if statement to remove 1 as prime
3b	Test the number range 5 to 600	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	None
3c	Test the number range 23 to 9,999	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	Worked and only took 2 seconds rather than 20 in V1
3d	Test the number range 300 to 7,908	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	Worked nearly instantly instead of 10 seconds
3e	Test the number range 89 to 104	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	None
3f	Test the number range 1,000 to 2,000	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	None
3g	Test the number range 2389 to 67,000	All primes in range print to screen	All primes in range print to screen	KL	✓	✓	✓	Only took 26 seconds instead of 3 minutes
4 REQUIREMENTS - Number must be a whole integer in feature 2								
4a	Input 1 fraction: 0.6 & 100	"Please ensure both numbers are greater than 1"	"Please ensure both numbers are greater than 1"	KL	✓	✓	✓	None
4b	Input 1 fraction box 2: 26 & 500.78	"Number entered must be a whole integer."	All primes in range print to screen	KL	✗	✗	✗	Incorporate bug fix to stop user entering a fraction in the 2nd box
4c	Input one fraction > 1 in box 1: 34.87 & 320	"Number entered must be a whole integer."	All primes in range print to screen	KL	✗	✗	✗	Incorporate bug fix to stop user entering a fraction in the 2nd box
4d	Input 2 fractions: 89.5 & 976.45	"Number entered must be a whole integer."	All primes in range print to screen	KL	✗	✗	✗	Incorporate bug fix to stop user entering a fraction in the 2nd box
5 REQUIREMENTS - Lower bound number must actually be lower than the higher bound number								
5a	Test numbers 100 & 3	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None
5b	Test numbers 1000 & 378	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None
5c	Test numbers 10 & 1	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None
5d	Test numbers 16,000 & 650	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None

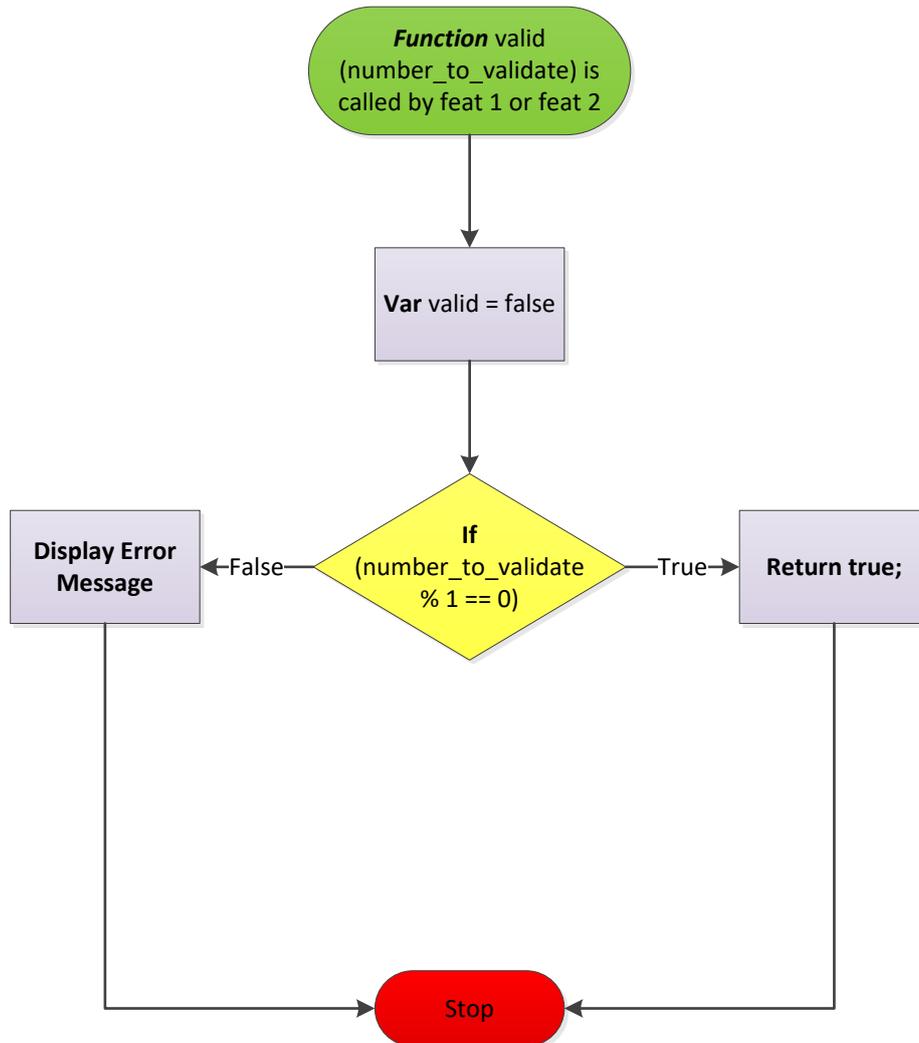
Optimisations & Bug fixes incorporated for V 3.0

No	Bug/Description	Fix
7	The number 1 was showing up as prime	Added an IF statement before the while loop to test for the no 1
8	Program would accept fractions and parse them to an integer	removed the parseInt around the number user inputs and replaced with parseFloat - also changed the test conditions to use % in the validate function for fractions. So if a fraction is entered, it would not be a modulus of 1 and an error message can be displayed
No	Optimisations	Result
9	All variables declared at the top of each function	The browser now reads these first - therefore more efficient
10	Feature 1 & 2 both use the same is_prime() function and valid() function	More efficient as less code
11	Made use of CSS instead of inline styling & HTML styling for page	CSS placed in header - More efficient as read first by the browser

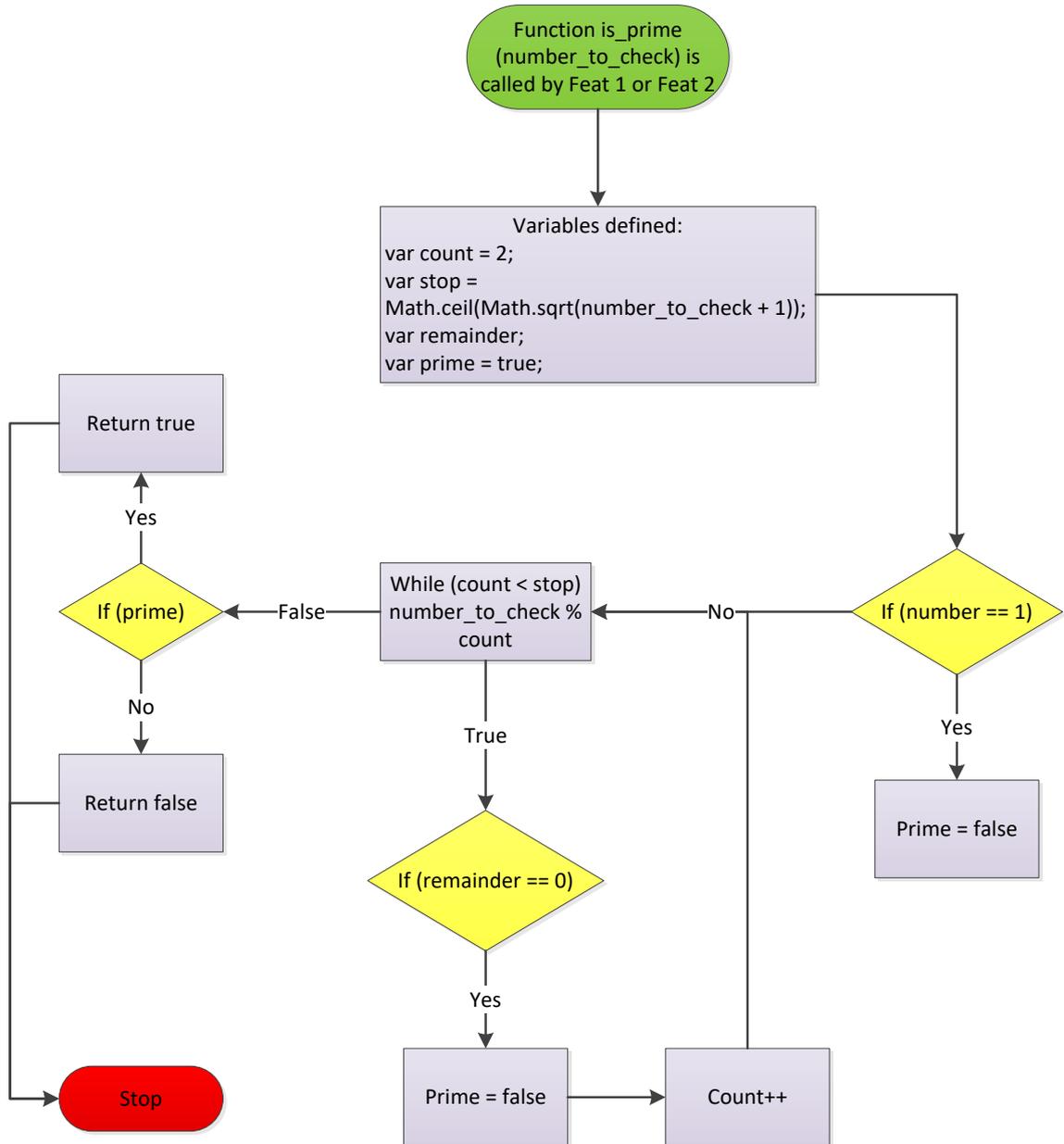
Overall Flowchart for V3



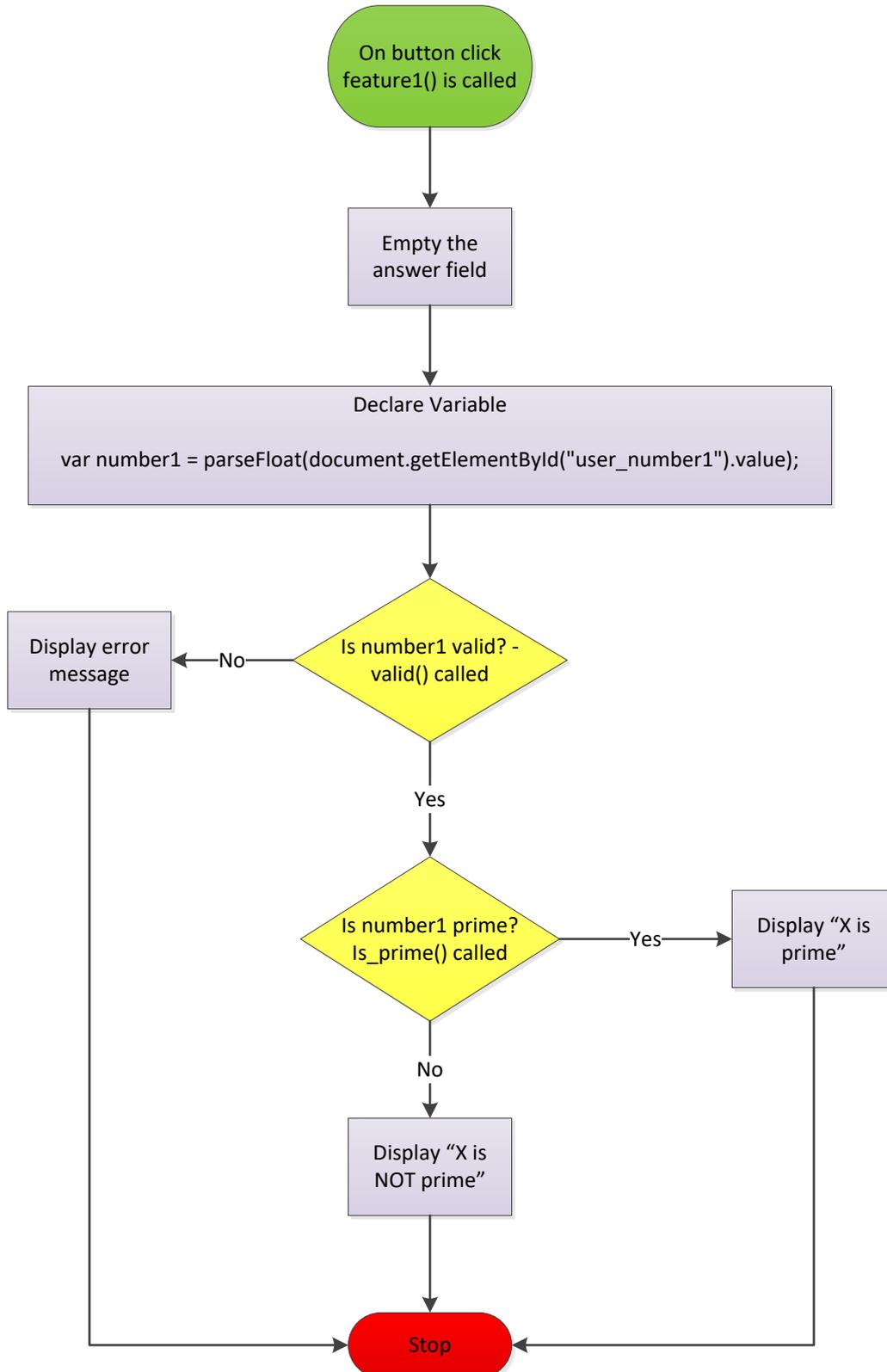
Function valid (number_to_validate) V3



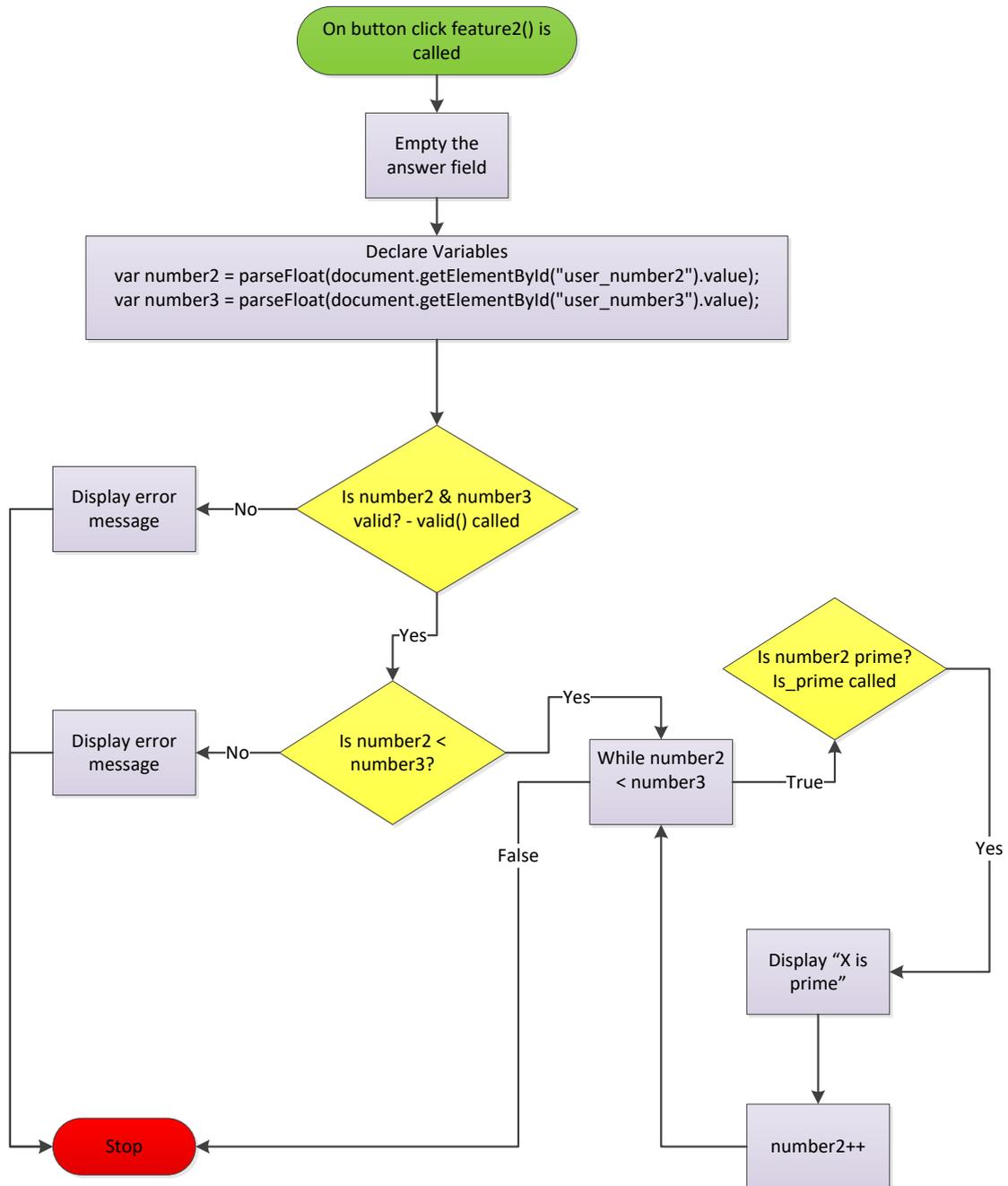
Is_prime() – Version 3 Flowchart



Function feature1() V3 Flowchart



Function feature2() V3 Flowchart



Version 3 Code - Feature 1 & 2

```

<!DOCTYPE html>
<html>
  <head>
    <title>Javascript Prime Number Checker</title>
    <style type="text/css">

      /* CSS Styling for page - user interface, including font type, size, padding
and borders*/
      * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        -webkit-box-sizing: border-box;
        -moz-box-sizing: border-box;
        -o-box-sizing: border-box;
        -ms-box-sizing: border-box;
        -webkit-text-size-adjust: none;
      }

      /*TITLES STYLING*/
      h1, .h1 {
        font-style:italic;
        font-size: 2.2em;
      }
      h2, .h2 {
        font-size: 1.6em;
      }
      h1,
      h2 {
        color: #C09;
      }

      h2.h1 {
        font-style:normal;
        text-transform: uppercase;
        font-size: 1.6em;
      }

      .plain-text {
        font-size: 15px;
        color: #000;
        font-weight: 100;
        display: inline;
        font-style: normal;
      }
      h1.plain-text:after {
        content: ' - ';
      }

      /*OBJECTS Styling*/
      body {
        background: #99CCFF;
        font-size: 15px;
        text-align: center;
        font-family: sans-serif;
      }

      .main-box {
        width: 95%;
        margin: 10px auto;
        background: #ffffcc;
        padding: 1%;
        border: 6px solid #CC0099;

```

```

    }
    .bordered {
        border: 1px solid #CC0099;
        padding: 40px;
    }
    .bordered + .bordered {
        margin-top: 15px;
    }
    .calculator {
        display: inline-block;
        vertical-align: top;
        width: 49.5%;
        text-align: left;
        height: 230px;
    }
    .calculator + .calculator {
        margin-left: 1%;
    }
    .calculator h2 {
        margin-bottom: 20px;
    }
    .calculator form {
        margin-top: 20px;
    }
    .dual-input {
        display: inline-block;;
    }
    .user-input + .user-input {
        margin-top: 5px;
    }
    label:hover {
        cursor:pointer
    }
    .calculator label:after {
        content: ':';
    }
    .calculator input {
        padding: 5px;
    }
    .answer {
        min-height: 130px;
    }
</style>
<script type="text/javascript">
    // validate function
    function valid(number_to_validate){
        //guilty until proven otherwise
        var valid = false;

        //Number entered must be a whole integer if modulus is not equal 0,
        then it's a fraction. Else, return true

        if (number_to_validate % 1 != 0){
            answer1.innerHTML = "Number entered must be a whole integer.";
            return false;
        }
        else{
            return true;
        }
    }

```

```

//check if prime
function is_prime(number_to_check){
    var count = 2; // start by testing the number against 2
    var stop = Math.ceil(Math.sqrt(number_to_check + 1)); //stop when count
has reached the sqrt of number rounded up + 1
    var remainder; // modulus test for remainder
    var prime = true; //start off by calling all numbers prime

    //1 is not prime
    if (number_to_check == 1){
        prime=false;
    }
    else{
        while(count < stop){
            //while count is less than stop, continue to test for a
remainder each time with the modulus operator
            remainder = number_to_check % count;
            if(remainder == 0){
                //If there is a remainder then prime = false
                prime=false;
                break;
            }
            //add one to count each time the program goes through the loop
            count++;
        }
        if (prime){
            return true;
        }
        else{
            return false;
        }
    }
}

//feature 1 program begin
function feature1(){

    //make sure the answer field is empty
    answer1.innerHTML = '';

    //Number entered into feature 1 form
    var number1 =
parseFloat(document.getElementById("user_number1").value);

    //Calls the valid function to check the number entered is valid
    if(valid(number1)){
        //Checks if the number is prime by calling the prime function
        if(is_prime(number1)){
            //Displays "X is prime" if prime function returns true
            answer1.innerHTML = + number1 + " is prime, ";
        }
        //Displays "X is NOT prime" is prime function returns false
        else {
            answer1.innerHTML = + number1 + " is <strong>NOT</strong>
prime, ";
        }
    }
}

//feature 2 prgram begin
function feature2(){

```

```

        //make sure the answer field is empty
        answer1.innerHTML = '';

        // Lower bound number entered into feature 2 form
        var number2 =
parseFloat(document.getElementById("user_number2").value);

        //Upper bound number entered into feature 2 form
        var number3 =
parseFloat(document.getElementById("user_number3").value);

        //Calling the valid function to check numbers are ok to be tested
        if(valid(number2) && valid(number3) ){
            //Tests if number2 is less than number 3
            if(number2 < number3){
                while(number2 < number3){
                    //Calls the is_prime function
                    if(is_prime(number2)){
                        //Displays "X is prime" if prime function returns true
                        answer1.innerHTML += + number2 + " is prime, ";
                    }
                    //Continues to print out prime numbers up to the higher
bound number entered
                    number2++;
                }
            }
            //Displays error message if number2 is higher than number3
            else {
                answer1.innerHTML = "The 2nd number must be higher than the
1st!";
            }
        }
    }

</script>
</head>
<body>
    <!--Begin HTML Layout-->
    <article class="main-box">
    <h1 class="bordered header">
        Javascript Prime Number Checker
    </h1>
    <section class="bordered calculator">
    <h2>
        Prime Number Calculator 1
    </h2>
    <p>
        Use this tool to calculate if a single number is prime.
    </p>
    <form>
        <label for="user_number1">
            Enter number
        </label>
        <input type="number" id="user_number1" placeholder="0"/>
        <input type="button" value="Calculate" onclick="feature1()"/>
    </form>
    </section><section class="bordered calculator">
    <h2>
        Prime Number Calculator 2
    </h2>
    <p>

```

Use this tool to find a range of primes between two given numbers.

```

</p>
<form>
  <div class="dual-input">
    <section class="user-input">
      <label for="user_number2">
        Enter Number 1 (Lower bound)
      </label>
      <input type="number" id="user_number2" placeholder="0"/>
    </section>
    <section class="user-input">
      <label for="user_number3">
        Enter Number 2 (Higher bound)
      </label>
      <input type="number" id="user_number3" placeholder="0"/>
    </section>
  </div>
  <input type="button" value="Calculate" onclick="feature2()" />
</form>
</section>
<section class="bordered answer">
  <h2 class="h1">
    Answer
  </h2>
  <p id="answer1"></p>
</section>
<footer class="bordered">
  <h1 class="plain-text">
    Introduction to programming
  </h1>
  <h2 class="plain-text">
    Assignment by Kayleigh Lamb
  </h2>
</footer>
</article>
</body>
</html>

```

Testing Schedule for V 3.0 - Feature 1

NOTE: All tests performed in Google Chrome version 23.0.1271.64 m , Firefox version 17.0 & Internet Explorer version 9.0.10 (ALL browsers were up to date at time of test 24/

Test No.	Description	Expected Outcome	Actual Outcome	Test by	Chrome Pass?	IE Pass?	Firefox Pass?	Action?
1 Check if prime in feature 1								
1a	Test the number 1	"1 is NOT prime"	"1 is NOT prime"	KL	✓	✓	✓	None
1b	Test the number 2	"2 is prime"	"2 is prime"	KL	✓	✓	✓	None
1c	Test the number 23	"23 is prime"	"23 is prime"	KL	✓	✓	✓	None
1d	Test the number 99	"99 is NOT prime"	"99 is NOT prime"	KL	✓	✓	✓	None
1e	Test the number 99,999	"99,999 is NOT prime"	"99,999 is NOT prime"	KL	✓	✓	✓	None
1f	Test the number 9007199254740992	"9007199254740992 is NOT prime"	"101 is prime"	KL	✓	✓	✓	None
2 Number must be a whole integer in feature 1								
2a	Test the number 0.2	"Number entered must be a whole integer."	"Number entered must be a whole integer."	KL	✓	✓	✓	None
2b	Test the number 2.9	"Number entered must be a whole integer."	"Number entered must be a whole integer."	KL	✓	✓	✓	None
2c	Test the number 100.6	"Number entered must be a whole integer."	"Number entered must be a whole integer."	KL	✓	✓	✓	None
2d	Test the word "rhubarb"	Screen clears text away	Screen clears text away	KL	✓	✓	✓	None
2e	Test the number 3478.56	"Number entered must be a whole integer."	"Number entered must be a whole integer."	KL	✓	✓	✓	None
2f	Test the number 2,300,456,9876	"Number entered must be a whole integer."	"Number entered must be a whole integer."	KL	✓	✓	✓	None

Testing Schedule for V 3.0 - Feature 2

NOTE: All tests performed in Google Chrome version 23.0.1271.64 m , Firefox version 17.0 & Internet Explorer version 9.0.10 (ALL browsers were up to date at time of test 24/11/2012)

**Reference list used to check my primes are correct can be found here: - <http://primes.utm.edu/lists/small/1000.txt>

Test No.	Description	Expected Outcome	Actual Outcome	Test by	Chrome Pass?	IE Pass?	Firefox Pass?	Action?
1 ** REQUIREMENTS - Check for a range of primes in Feature 2								
1a	Test the number range 1 to 100	All primes between range print to the screen	All primes between range print to the screen	KL	✓	✓	✓	None
1b	Test the number range 5 to 600	All primes between range print to the screen	All primes between range print to the screen	KL	✓	✓	✓	None
1c	Test the number range 23 to 9,999	All primes between range print to the screen	All primes between range print to the screen	KL	✓	✓	✓	None
1d	Test the number range 300 to 7,908	All primes between range print to the screen	All primes between range print to the screen	KL	✓	✓	✓	None
1e	Test the number range 89 to 104	All primes between range print to the screen	All primes between range print to the screen	KL	✓	✓	✓	None
1f	Test the number range 1,000 to 2,000	All primes between range print to the screen	All primes between range print to the screen	KL	✓	✓	✓	None
1g	Test the number range 2389 to 67,000	All primes between range print to the screen	All primes between range print to the screen	KL	✓	✓	✓	None
2 REQUIREMENTS - Number must be a whole integer in feature 2								
2a	Input 1 fraction: 0.6 & 100	"Number entered must be a whole integer."	"Number entered must be a whole integer."	KL	✓	✓	✓	None
2b	Input 1 fraction box 2: 26 & 500.78	"Number entered must be a whole integer."	"Number entered must be a whole integer."	KL	✓	✓	✓	None
2c	Input one fraction > 1 in box 1: 34.87 & 320	"Number entered must be a whole integer."	"Number entered must be a whole integer."	KL	✓	✓	✓	None
2d	Input 2 fractions: 89.5 & 976.45	"Number entered must be a whole integer."	"Number entered must be a whole integer."	KL	✓	✓	✓	None
3 REQUIREMENTS - Lower bound number must actually be lower than the higher bound number								
3a	Test entering 81 & 2	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None
3b	Test entering 4567 & 56	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None
3c	Test entering 675 & 106	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None
3d	Test entering 981024 & 256	The 2nd number must be higher than the 1st!	The 2nd number must be higher than the 1st!	KL	✓	✓	✓	None

Summary

Version 1, with both feature 1 & 2 as separate programs, gave me a chance to get to know the requirements and test out different ideas.

Versions 2 & 3 allowed me to refine my ideas and where necessary, re-design small parts of my code. Throughout this document I have tried to demonstrate the journey taken to get to Version 3.

Version 2, saw the programs concatenated and making use of some of the same functions. In Version 3, the fractions bug was fixed along with the "1 is not prime" issue.

My main criticism of the final program is that it takes a long time to find a range of primes when the second number is very big. Although the efficiency is increased when only testing up to the square root of a number, with REALLY big numbers, it can still take a long time to return the results. For the users of this program, I would advise using feature 1 only to check if a big number is prime.

To resolve this issue in future versions, I would look at a method that somehow breaks the big number up before testing.

Lastly, other than the big numbers issue, I believe that my program meets all of the requirements and that I have provided sufficient evidence to back this up.