

```

if ( _____ )
    alert("Odd number");
else
    alert("Even number");

```

29. What is the value of the following, given that `myName = "M. Nguyen III"`?

a) `myName.charAt(3)` b) `myName.charAt(10)`

30. Add the condition needed to allow the player of a guessing game to make another guess for a secret number. Another guess is allowed if the player has previously guessed incorrectly and has not used 10 guesses.

```

var secret = Math.floor(Math.random()*10);
var numGuess = 1;
var newGuess = parseInt(prompt("Take a guess: ", " "));
while ( _____ )
{
    alert("incorrect");
    newGuess = parseInt(prompt("Guess again: ", " "));
    numGuess++;
}

```

Programming Challenges

On Your Own

1. Create a web page that will display a countdown to a rocket blasting off. Use a button to start the countdown. The countdown sequence should display on the web page and should display **BLASTOFF!** at the end. You can, if you wish, add a rocket image. Save the page as `blastoff.html` and be sure to include an appropriate page title.
2. Create a web page that asks the user for a password. The password must have exactly eight characters and may not include spaces. All other keyboard characters are allowed. A loop should prompt the user to re-enter another password until both these conditions are met. Save your page with the filename `password.html` and be sure to include an appropriate page title.
3. Create a web page that allows the user to evaluate his or her car's performance on various trips. The user should be allowed to enter as many data sets as desired. For each set the following information will be entered: the name of the trip, the number of miles driven, and the number of gallons of gas used. The output should be put into a JavaScript-generated table and look like this:

Trip Name	Miles Driven	Gallons Used	Miles per Gallon
Disney World	560	20	28 mpg
...
...
New York City	152	8	19 mpg

Save your page as `mpg.html` and be sure to include an appropriate page title.

4. The factorial of a number, N , is defined as follows:

$$N! = 1 * 2 * 3 * 4 * \dots * N$$

For example, $4! = 4 * 3 * 2 * 1$ and $7! = 7 * 6 * 5 * 4 * 3 * 2 * 1$

Create a web page that allows the user to enter a positive integer and the page will display the factorial of that number. *Hint:* Use a variable, `factorial`, with an initial value of 1. Then use a loop to multiply `factorial` by successive integers up to the value entered by the user. Save your page as `factorial.html` and be sure to include an appropriate page title.

5. A biologist has determined that the approximate number of bacteria in a culture after a given number of days is given by the following formula:

$$\text{bacteria} = \text{initialBacteria} * 2^{(\text{days}/10)}$$

where `initialBacteria` is the number of bacteria present at the beginning of the observation period. Let the user input the value for `initialBacteria`. Then compute and display the number of bacteria in the culture over 10 days. Do this in a loop that also generates the output in a table on the web page as shown:

Initial Bacteria present:

Day	Bacteria
-----	----------

1	
---	--

.	
---	--

.	
---	--

10	
----	--

Save your page as `bacteria.html` and be sure to include an appropriate page title.

6. Create a web page that uses a loop to allow a teacher to enter the following information for all students in a class: student's name, midterm exam grade, final exam grade, homework grade, attendance grade. The program should calculate each student's numeric total grade based on the following formula:

$$\text{grade} = (\text{midterm} * 0.3) + (\text{final} * 0.4) + (\text{homework} * 0.2) + (\text{attendance} * 0.1)$$

Display the results in a JavaScript-generated table as shown:

Name	Attendance	Homework	Midterm	Final	Course Grade
Bianca	90	85	78	92	86.2
.
.
Walter	70	50	83	74	71.5

Save your page as `points.html` and be sure to include an appropriate page title.

7. Create a web page that allows a player to enter a message. When a button is clicked, all the text on the page (except the text on the button), including the player's message, will be displayed in reverse. For example, if the page had a

heading that says **Mirror Image**, after the button is clicked, the header will say **egamI rorriM**. The player should also be able to return the page to normal. *Hint:* What would happen if you reversed a word that had been reversed? Save your page as `mirror.html` and be sure to include an appropriate page title.

Case Studies

Greg's Gambits

Here you will add to the encryption page created earlier in this chapter. The encryption system that has been written in this chapter would be pretty easy for someone to figure out. Now you will create a more sophisticated encryption algorithm.

Open the `play_games.html` page and add a link, under the **The Secret Message Encoder** link that links to the new page you will create. The new page should have **Unbreakable Secret Message Encoder** as a page title and the filename should be `gregs_encoder2.html`.

You can use the page created earlier in the chapter as a template. The page title, **Unbreakable Secret Message Encoder**, should also be the first header on the page. In the content area, place a button that the player can click when he or she is ready to encrypt a message.

In the chapter example, we used the following algorithms to encode text:

- uppercase letters: `newcode = (upCaseCode - msg.charCodeAt(j));`
- lowercase letters: `newcode = (lowCaseCode - msg.charCodeAt(j));`
- numbers and special characters: `newcode = (msg.charCodeAt(j) + specialCode);`

where `newcode` was the new character in the encoded message, `msg` was the message the user entered, `j` was the index of the character in question, `upCaseCode = 155`, `lowCaseCode = 219`, and `specialCode = 3`.

Now you will generate a random number that will move the value of the `newcode` up (or down) by a certain number of characters. By generating a new random number each time a message is encrypted, the code will be a lot more difficult to break. In a real encryption program, this number would be sent to both the player who is encrypting the message and the person who is receiving the message so it can be decrypted. It would be the key. But anyone else who sees the secret message and does not have the key will not know how to decrypt it. On our page we will just ask the player if he wants to see the random number that has been used in the encryption. If yes, then display that number along with the encryption algorithm. Use the table of Unicode values to ensure that you do not generate any characters outside the range of acceptable ASCII keyboard strokes.

For example, you might use the following for encoding uppercase values: There are 26 letters in the alphabet and uppercase values range from 65 to 90 while lowercase values range from 97 to 122. You might decide to change an uppercase A to its lowercase value plus a random number between 2 and 26. If the random number was 5, this would encode A to Unicode 97 (lowercase a) plus 5 or Unicode 102 which

is lowercase f. Using this scheme, the word CAT would become hfy. You need to consider, of course, what would happen to an uppercase Z? The Unicode value for z is 122; therefore, $122 + 5 = 127$ which is not included in the ASCII values we can use. You will need to add code to account for this situation. How you do this is up to you. You could simply define all characters that are greater than 126 to be a specific character. Or you could alter your algorithm to change the addition to subtraction in this case. You may think of another clever way to control this situation.

For this program, create new algorithms for uppercase and lowercase letters as well as digits and punctuation. Test your page in at least two different browsers. Submit your work as instructed by your teacher.

Carla's Classroom

Now you will add to the addition and subtraction exercises created earlier in this chapter by creating multiplication and division exercises.

Each page or exercise you create should prompt the student to answer an arithmetic question. The student should be given the option to continue until five questions are answered correctly or to stop after a desired number of questions. As with the addition and subtraction exercises created in the chapter, each question should be created by generating two random numbers. The correct answer should be compared to the student's answer. There should be two counters: one to count the number of correct answers and one to count the number of problems attempted. At the end of each level, the student should receive a message stating how many questions were attempted and how many were correct. At the end of each level the student should be prompted to continue to another level, if one is available, or to end the program.

Open the math.html file and add one or more links (depending on how many options you choose to program) under the Advanced Addition and Subtraction Exercises. The links should go to the new page or pages you create.

Use the carla_math_whiz.html page that was created earlier in the chapter as a template. You can create one or both of the following arithmetic exercises:

- Create two levels of multiplication. The first level should multiply whole numbers between 1 and 10 and the second level should multiply numbers between 10 and 100. You can add this code to the carla_math_whiz.html page created earlier in the chapter or, if you only do this part, name this page adv_multiplication.html.
- Create two levels of division problems. The first level should divide whole numbers between 1 and 100 but only generate problems that result in integer quotients. The second level should also include integers between 1 and 100 but allow for decimal responses (i.e., $11/2 = 5.5$). You need to make sure your code checks for the following:
 - For the first level, the dividend must be bigger than or equal to the divisor; i.e., in the expression $a \div b$, a must be larger than (or equal to) b.
 - For the second level, instruct the student to display the answer only to two decimal places and be sure your result is also truncated after two decimal places.




Chapter Review and Exercises

You can add this code to the `car1a_math_whiz.html` page created earlier in the chapter or, if you only do this part, name this page `adv_division.html`.

Test your page(s) in at least two different browsers. Be sure to test all possible combinations of correct and incorrect responses at each level. Submit your work as instructed by your teacher.

Lee's Landscape

In this exercise, you will use the table of services from the Lee's Landscape | Services page that you created in Chapter 3. On this new page, the customer will order services. The page should prompt the user to enter an amount he or she wishes to spend. Then the customer can order services from the table of services (copied from Chapter 3, below, for your convenience). The program should keep a running total of how much the customer has spent after each entry. If the customer tries to go over his initial spending limit, the customer should be alerted to this fact; the last item ordered should be deleted from the list; and the customer prompted to either order something more affordable or stop ordering. The final output should be a table that lists the services the customer has ordered, the prices, and a grand total.

Service	Options	
Lawn maintenance	Weekly: \$15/service Twice a month: \$25/service Monthly: \$40/service	
Pest control	Monthly: \$35/service Twice a year: \$75/service Yearly: \$150/service	
Tree and hedge trimming	Monthly: \$25/service Twice a year: \$75/service Yearly: \$150/service	

The images used here are available in the Student Data Files. Of course, you can substitute your own images if you wish. Be sure to give this web page an appropriate page title; Lee's Landscape | Order Services is suggested. Save this file with the filename `lee_order.html`. Add a link to the Lee's Landscape home page (if you created one in a previous chapter) to this new page. Test your page in at least two different browsers. Be sure to test all possible combinations of each service and contract. Submit your work as instructed by your teacher.

Jackie's Jewelry

Add a page to the Jackie's Jewelry website that allows a customer to select free samples. Jackie offers eight possible samples from which a customer can pick and gives free samples as follows:

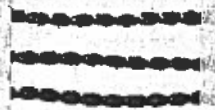
- If the purchase is any amount up to (and including) \$50.00, one sample is given.
- If the purchase is between \$50.01 and 100.00, two samples are given.
- If the purchase is over \$100.00, three samples are given.

Later in the text you will create a shopping cart for Jackie. However, at this point, you can simply prompt the customer to enter the amount of his or her purchase. The program should then tell the customer how many samples can be chosen and ask the customer to pick from the following list. Then the program should display the samples the customer will receive.

Available Samples



two glass beads



two yards satin cord (pink)



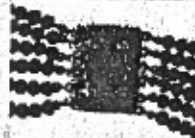
velvet ring box



one yard leather cord



bracelet box



one clasp



package of assorted beads



pendant

The images used here are available in the Student Data Files. Of course, you can substitute your own images if you wish. Be sure to give this web page an appropriate page title; Jackie's Jewelry | Samples is suggested. Save this file with the filename `jackie_samples.html`. Add a link to the Jackie's Jewelry home page to this new page. Test your page in at least two different browsers. Submit your work as instructed by your teacher.