# MIS2402 – Hands On Prep

## Part A – Multiply the natural numbers between 1 and x.

1. Download prep.zip to your computer.
2. Working in prepa.html found in the prep folder, complete the function getFactorial.
	* You should assume that the user has entered an integer. Don't worry about checking for non-numeric user input, or for user input with decimal precision.
	* If the user entered a negative number, the function should return the string "error"
	* If the user entered a zero, then the function should return the number: 1
	* If the user enters a number of 1 or higher, return the product of the numbers between 1 and x. For example, if the user entered 4, the function should return 24 because 1$×$2$×$3$×$4 equals 24.
	* HINT: You will need a for loop to solve this problem.
3. Check your work.
4. Fun fact: In mathematics, the product of all the natural numbers between 1 and x is also called "x factorial". That's why the function is called getFactorial.

## Part B – Funky Factorial.

1. Working in prepb.html found in the prep folder, complete the function getFunkyFactorial.
	1. Start with your solution to the last problem
	2. Inside your loop, add an “if” statement to see if the counter is a multiple of 3. If the counter is a factor of three, multiply it by all previous factors of three. If it’s not a factor of three, go on to the next value of x.

HINT: Use the modulus (%) operator to determine if the counter % 3 is zero or not.

NOTE: You could have done this by iterating through the loop in steps of three. However, it is important for students to observe how “if” statements can be embedded inside of a loop. This is also good practice using the modulus operator!

1. Check your work.
	* The Funky Factorial of 10 would be 162 because: 3 x 6 x 9 = 162
	* The Funky Factorial of 15 would be 29160 because: 3 x 6 x 9 x 12 x 15 = 29160

## Part C – Using loops to generate text (OPTIONAL)

1. Working in prepc.html found in the prep folder, complete the solution as instructed:
	1. In this part, we ***won’t assume*** that the user has entered an integer.
	2. This time, we will put our error trap for non-natural user input into the ***event handler***.
	3. If the user gave you bad input, put “Bad data. Try again.” into the div tag textDisplayed1
	4. If the user gave your good input, call the makeStaircase function and put the output of that function into textDisplayed1
	5. Complete the function makeStaircase. It should produce a string of “X” characters as and “<br>” tags as shown in the illustrations below.
	6. HINT: To solve this problem, you will need a for loop inside another for loop.

 

CONTINUED

 

1. Check your work
2. In this example, we see that error trapping may occur in the event hander. You have also seen error trapping in the function itself. *Always read all quiz and assignment instructions carefully to determine where the error trapping is expected, and where it is not!*
3. In this example, we also saw that a loop counter (i) does not always need to be part of a mathematical formula. Sometimes, it just controls the loop.