# MIS2402 – Assignment 05: Improved Treadmill Simulation

## Scenario

In this assignment you will expand on our basic treadmill simulation. The original program burned 148 calories every 10 minutes and stopped after one hour. To make the exercise more useful we will let the user enter the length of the workout and the calories burned every 10 minutes. The program will show progress after each 10‑minute interval. This is still a very simple simulation; a real treadmill would adjust for height, weight and the difficulty setting.

You will complete three small functions that show how loops, input validation and return values work together. Each button on the web page runs a slightly more sophisticated version of the treadmill calculation.

## Step‑by‑Step Instructions

1. **Log in to the class server.** Use the username and password your instructor gave you. The server’s address is misdemo.temple.edu. After you log in, click the **New terminal console** icon in Bitvise.
2. **Download and prepare the assignment.** At the prompt, type each of these commands exactly as shown (use Courier New font when typing commands):

cd wwwroot

wget https://misdemo.temple.edu/assignments/assignment05.zip

unzip assignment05.zip

rm assignment05.zip

cd assignment05

1. **Open the start file in your browser.** Replace <yourusername> with your own account name:
* https://misdemo.temple.edu/<yourusername>/assignment05/improvedtreadmill.html
* You should see a form with two text boxes and three buttons labeled **Treadmill A**, **Treadmill B** and **Treadmill C**. If so, you are ready to begin.
1. **Open the HTML file for editing.** Return to your terminal and open the page with nano:

nano improvedtreadmill.html
* Scroll down to the <script> tag near the bottom. You will see three empty functions: treadmillA, treadmillB and treadmillC. Write your code inside these functions. **Do not change the names or parameters.**
1. **Complete treadmillA().** This function has no parameters and always uses the literal values of 60 minutes and 148 calories per 10 minutes. (Just like in the class activity)
	* Create a variable named totalcalories and set it to 0.
	* Create a variable named progress and set it to an empty string ("").
	* Write a for loop that starts at 10 and runs up to 60, counting by 10.
	* Inside the loop, add **148** to totalcalories and append the new total as well as a comma to progress.
	* Use an if statement so that a comma and space are \*not\* added after the last number. For example, the final value of progress should be "148, 296, 444, 592, 740, 888".
	* At the end of the function use return progress;.
	* Save your file and test in the Chrome developer console by typing treadmillA();. The function should return the string of totals.
2. **Complete treadmillB(duration, calorierate).** This function uses the values typed into the form. Remember that text inputs are strings. Any data you collect off an HTML form is of the datatype string! That means that, sometimes, we need to deliberately convert a string to a number using a function like parseInt or parseFloat.
	* At the start of the function cast both parameters to numbers using parseFloat().
	* Validate the inputs. If either value is not a number (isNaN() returns true) or is less than 0, immediately return "Bad data. Try again".
	* If the data is good, set totalcalories to 0 and progress to an empty string.
	* Write a for loop that starts at 10 and runs up to the value of duration, counting by 10. On each iteration add calorierate to totalcalories and append the new total to progress.
	* Use if (i < duration) to decide whether to add a comma and space after the number.
	* Return progress.
	* Test your function using the browser console. For example:
		+ treadmillB(20, 100) should return "100, 200".
		+ treadmillB(30, 200) should return "200, 400, 600".
3. **Complete treadmillC(duration, calorierate).** This function is similar to treadmillB() but returns HTML for a list.
	* Cast and validate the parameters using the same logic as in step 6. Invalid input should return "Bad data. Try again".
	* Create a string called progress that starts with <ul><li>.
	* Use a for loop to add calorierate to the running total at each 10‑minute interval.
	* Append each new total inside a <li> tag. If i < duration, close and open the next list item (</li><li>). Otherwise close the list (</li></ul>).
	* Return the HTML string. For example, treadmillC(20, 100) should return <ul><li>100</li><li>200</li></ul>.
4. **Save and test your code.** Press **Ctrl‑O** to save in nano and refresh your browser page. Click each button to verify that the correct result appears. Open the developer console and run treadmillA(), treadmillB(20, 100) and treadmillC(30, 150) to verify that the returned values are correct and that invalid input shows the error message.

CONTINUED

1. Determine the **URL** to your improved treadmill page. It will look like:
* https://misdemo.temple.edu/<yourusername>/assignment05/treadmill.html
* Replace <yourusername> with your own account and test the link to make sure it works.
1. Go to the assignment in Canvas and paste your URL into the submission field. Be sure your page loads without errors.

## Facts & Formulas

* This simulation assumes a constant calorie burn per ten‑minute interval. A real treadmill would take more factors into account such as the runner’s height, weight and the difficulty level setting on the treadmill itself.
* Use parseFloat() to convert strings to numbers when reading values from text inputs.
* Use isNaN() to determine whether a value is not a valid number.
* A for loop is ideal for repeating actions. In these functions the loop counter starts at 10 and increases by 10 in each iteration.
* To avoid a trailing comma at the end of your progress string, compare the loop counter to the total duration (i < duration).

## Hints & Reminders

* Always begin a function with parameters by validating the inputs. Return the error message immediately if the data is bad.
* Use return to send a result back to the caller. Do not rely on console.log() for the final answer.
* Keep your functions focused on one task.
* When creating HTML strings, remember to include the opening and closing tags in the correct order (<ul> <li> </li> </ul>).
* Test your functions often in the browser console. Use simple values like treadmillB(20, 100) and treadmillC(30, 150) to check your logic.

## Grading rubric:

* **100 points:** You provided a valid URL, all three functions work correctly, and your code follows the rules.
* **80 points:** Minor problems such as a small miscalculation or formatting issue.
* **50 points:** Multiple problems but the page does not crash and you attempted the assignment.
* **0 points:** The file is missing, the URL is wrong, or the code uses features not covered in class.