Algorithm Documentation: Binary Search-based Gold Bar Identification

Overview:
This algorithm aims to identify a fake gold bar among a set of genuine bars using a binary search approach. The identification process involves repeatedly comparing the weights of the gold bars until the fake bar is found. This algorithm is implemented within a Selenium-based automation framework for web testing.

Algorithm Steps:

Initialization:

- Define an array of weights containing the numbers representing the gold bars.
- Initialize variables l and r representing the left and right indices of the current search range, respectively.

Binary Search Iteration:

- Enter a loop that continues until the left index l is less than or equal to the right index r.
- Compute the next l and r depending on the bowl we have to search in.

Weighing Process:

- Click the "Reset" button to clear any previous weighing.
- Fill the left and right bowls with the subsets of gold bars based on the current search range using the binary search approach.
- Click the "Weigh" button to compare the weights of the left and right bowls.

Result Analysis:

- If the weights are equal (=), the fake gold bar is identified, and the corresponding action is taken.
- If the weight on the left side is less than the weight on the right side (<), update the search range to the left half by setting r to middle value - 1.
- If the weight on the left side is greater than the weight on the right side (>), update the search range to the right half by setting l to middle value + 1.

Repeat:

- Repeat steps 2-4 until the fake gold bar is found.

Time Complexity:

- The time complexity of the algorithm is O(log n), where n is the number of gold bars. This is because the algorithm halves the search range in each iteration, similar to a binary search.

Space Complexity:

- The space complexity of the algorithm is O(1) because it only requires a constant amount of additional space for variables such as l, r. The weights array, although containing n elements, is reused across iterations and does not contribute to the space complexity within the loop.

Implementation Details:

- The algorithm is implemented within a Selenium-based automation framework, allowing interaction with a web application to simulate the weighing process and identify the fake gold bar.

Additional Challenge Function: Displaying Weighing List Output

Function showWeighingList(driver) has been implemented to print the weighing list on the output console. This function provides visibility into the history of weighing operations performed during the gold bar identification process.

Additional Test Cases:

Multiple test cases have been created to test various UI functions of the web application. These test cases cover scenarios such as button functionality, measurement results accuracy, bowl filling conditions, and bottom gold bar button interactions. To run these test cases, uncomment the relevant sections in the code and execute them.