**Task 1: Implement Bubble Sort with Optimization**

**Description:**  
Write a Python function that implements the **optimized Bubble Sort** algorithm. The optimization should include a **flag**that stops the algorithm if the array is already sorted before completing all iterations.

**Requirements:**

* Implement the **optimized** Bubble Sort algorithm.
* The function should take a **list of integers** as input.
* Print the list after **each pass** of the algorithm to show the sorting progress.
* If a pass occurs without a swap, terminate the function early.

**Example Input:**

arr = [5, 1, 4, 2, 8]

**Example Output:**

Pass 1: [1, 4, 2, 5, 8]

Pass 2: [1, 2, 4, 5, 8]

Pass 3: [1, 2, 4, 5, 8] (No swaps, terminating early)

Sorted Array: [1, 2, 4, 5, 8]

**Task 2: Count and Compare Swaps in Bubble Sort vs. Insertion Sort**

**Description:**  
Write a Python program that:

1. Sorts an array using **Bubble Sort** and counts the **total number of swaps** performed.
2. Sorts the same array using **Insertion Sort** and counts the **total number of swaps** performed.
3. Compares the number of swaps in both sorting algorithms.

**Requirements:**

* Implement **Bubble Sort** and **Insertion Sort**.
* Count and display the total **number of swaps** for both algorithms.
* Compare which algorithm performs **fewer swaps** for a given dataset.

**Example Input:**

arr = [7, 3, 8, 2, 5]

**Example Output:**

Bubble Sort Swaps: 7

Insertion Sort Swaps: 5

Insertion Sort performed fewer swaps.