**THE GRID SEARCH**

The Counting Sort algorithm is an efficient, non-comparison-based sorting technique. It works by counting the number of occurrences of each unique element in the input array and then arranging them in the correct order. Unlike traditional sorting algorithms that repeatedly compare elements, Counting Sort uses the frequency of elements to determine their positions, which makes it particularly effective when the range of values is limited.

In this implementation, the input consists of n elements, each represented as a pair of values. The program reads these inputs into an array, preparing them for sorting. The countSort function is intended to organize these elements based on their numeric value while maintaining the relative order of elements with the same key — a property known as stability. For the first half of the array, the algorithm may replace the string part with a placeholder like "-", and in the second half, it preserves the original strings, ensuring that the sorted output reflects the input order accurately.

Although the function body of countSort is currently empty, the algorithm typically involves creating a counting array where the index represents the key value, populating it with elements accordingly, and then reconstructing the sorted array by sequentially reading through the counting array. This approach allows sorting in linear time relative to the number of elements and the maximum key value.

By leveraging Counting Sort, we can efficiently handle sorting problems that involve a large number of elements with constrained key ranges, making it a practical choice for many real-world applications such as processing labeled data or preparing structured outputs.

**Implemented and explained by:**  
Mark Anthony Alava  
BSIT-3 Nexus