Ahmet Ulucay Drag-and-Carry Multiplication Algorithm (AUD&CMA v2)

-- Convolutional Form --

The Ahmet Ulucay Drag-and-Carry Multiplication Algorithm (AUD&CMA v2) introduces a novel approach to digit-wise multiplication by combining structured positional padding, convolution-like window sliding, and a specialized right-to-left carry mechanism. Unlike traditional methods that rely on place value decomposition or lattice grids, AUD&CMA v2 applies a reversed-digit kernel (from the multiplier) over the padded multiplicand, computing localized dot-products across digit windows.

This hybrid method integrates discrete convolution principles with mental-math-friendly carry handling, where all digit sums are reduced to single digits via right-to-left propagation, except for the leftmost value, which is preserved in full. The algorithm's unique structure and deterministic steps distinguish it both theoretically and practically from classical multiplication strategies.

Step-by-step Algorithm:

1. Convert inputs to digit arrays:
   ```
   N = list(map(int, str(n)))
   M = list(map(int, str(m)))
   ```

2. Reverse the multiplier for convolution:
   ```
   M = M[::-1]
   k = len(M)
   ```

3. Pad the multiplicand with k-1 zeros on both sides:
   ```
   pad = [0] * (k - 1)
   N_padded = pad + N + pad
   ```

4. Slide a window of size k over N_padded and compute dot-products:
   ```
   result = []
   for i in range(len(N_padded) - k + 1):
       window = N_padded[i:i+k]
       dot = sum(window[j] * M[j] for j in range(k))
   ```

```
        result.append(dot)

5. Apply drag-and-carry operation right-to-left:
   for i in range(len(result) - 1, 0, -1):
        carry = result[i] // 10
        result[i] = result[i] % 10
        result[i - 1] += carry

   # Note: Leftmost digit is preserved in full form.

6. Concatenate final result:
   final_number = int(''.join(map(str, result)))
```