Propose a Dynamic Programming algorithm that on input w, u, v checks whether $w = u \odot v$; that is, an array of subproblems, a recurrence, a pseudocode solution **(15 points)** and finally a Python 3 implementation **(5 points)**.

Consider $u = u_1u_2u_3...u_m$, $v = v_1v_2v_3...v_n$, and $w = w_1w_2w_3...w_{m+n}$, $|w| = |u| + |v|$

(i) Define a class of subproblems.

Solution: Define $S(i, j)$ to be true if the first $i + j$ characters in w are a shuffle of the first i characters of u and the first j characters of v. False otherwise.

(ii) Give a recurrence based on solving each subproblem in terms of simpler subproblems.

Solution: For any $0 \leq i \leq m$ and $0 \leq j \leq m$ we have:

$$S(i,j) = \begin{cases} \text{True} & \text{if } i = j = 0 \\ \text{True} & \text{if } i > 0 \text{ and } u_i = w_{i+j} \text{ and } S(i-1,j) = \text{True} \\ \text{True} & \text{if } j > 0 \text{ and } v_j = w_{i+j} \text{ and } S(i,j-1) = \text{True} \\ \text{False} & \text{otherwise} \end{cases}$$

(iii) Give a psuedocode algorithm for computing the recurrence.

Algorithm 1 Shuffle Algorithm

```
1: S(0,0) ← True
2: for i : 0..m do
3:     for j : 0..n do
4:         if i > 0 and u_i = w_{i+j} and S(i−1, j) = True then
5:             S(i, j) ← True
6:         else if j > 0 and v_j = w_{i+j} and S(i, j−1) = True then
7:             S(i, j) ← True
8:         end if
9:     end for
10: end for
11: return S(m, n)
```

(iv) Implement the algorithm in Python 3.

Solution: An example Python 3 solution can be seen in `a3-sol-comp554-w18.py`