EECS 336 Problem 6.2

1. Identify subproblem

Let p = a.length, q = b.length, r=c.length

OPT(i, j, k) = whether $\{c_k, c_{k+1}, ..., c_r\}$ could be a mixture of $\{a_i, a_{i+1}, ..., a_p\} \cup N * a$ and $\{b_j, b_{j+1}, ..., b_q\} \cup M * b$. (where M, N are nonnegative integers)

2. Recurrence:

OPT(i, j, k) = [OPT (i+1, j, k+1) and $c_k == a_i$] or [OPT(i, j+1, k+1) and $c_k == b_j$] (Note: if i = p, replace i+1 by 1; if j = q, replace j+1 by 1)

3. Base case:

1. OPT(1, 1, r+1) = True
2. OPT(i, j, r+1) = False (if not (i==1 and j==1))

4. Iterative Dynamic Programming

Algorithm 1 FiEncournstters

Require: two base strings a and b, candidate mixture array c

```
1: function FiEncournstters(a, b, c)
2:     Set p = a.length; q=b.length; r=c.length;
3:     Initialize Memo = new array[p][q][r+1]
4:     Set base case:
5:     Memo[1][1][r + 1] = True
6:     for i = 1 to p do
7:         for j = 1 to q do
8:             if i ≠ 1 or j ≠ 1 then
9:                 Memo[i][j][r+1] = False
10:            end if
11:        end for
12:    end for
13:    for k = r down to 1 do
14:        for i = 1 to p do
15:            for j = 1 to q do
16:                Memo(i, j, k) = [Memo (i+1, j, k+1) and c_k == a_i] or [Memo(i, j+1, k+1) and c_k == b_j] (Note: if i = p, replace i+1 by 1; if j = q, replace j+1 by 1)
17:            end for
18:        end for
19:    end for
20:    return Memo[1][1][1]
21: end function
```

RunTime

1. Initialization takes O(pqr) time to generate 3D matrix
2. Base case takes O(pq) time

3. Memo[ ][ ][ ] is a three-dimensional matrix. Each iteration (update) takes constant time. Total runtime is O(pqr)
So the total runtime is O(pqr).

Correctness of the recurrence
(Proof by induction)
Assume the boolean value of OPT(i, j, k) could represent whether $\{c_k, c_{k+1}, ..., c_r\}$ could be a mixture of $\{a_i, a_{i+1}, ..., a_p\} \cup N * a$ and $\{b_j, b_{j+1}, ..., b_q\} \cup M * b$. (where M, N are nonnegative integers).
Upon this, we will consider which base string c_{k-1} should belong to. c_{k-1} could belong to a when $c_{k-1} = a_{i-1}$. Then by taking c_{k-1} into consideration, OPT(i-1,j,k-1) = OPT(i,j,k) and $c_{k-1} = a_{i-1}$. Another possible scenario is that c_{k-1} is treated as part of b. Similarly, OPT(i, j-1, k-1) = OPT(i,j,k) and $c_{k-1} = b_{j-1}$. When either of the two scenarios happens, the property of OPT holds, So there should an OR relation between the two cases. Therefore, the recurrence holds.