Draw Proof tree for Query :
? member (3, [1, 2, 3, 4])

Member (3, [1, 2, 3, 4])

false   member (3, [2, 3, 4])

false   member (3, [3, 4])

true
Since 3 = 3
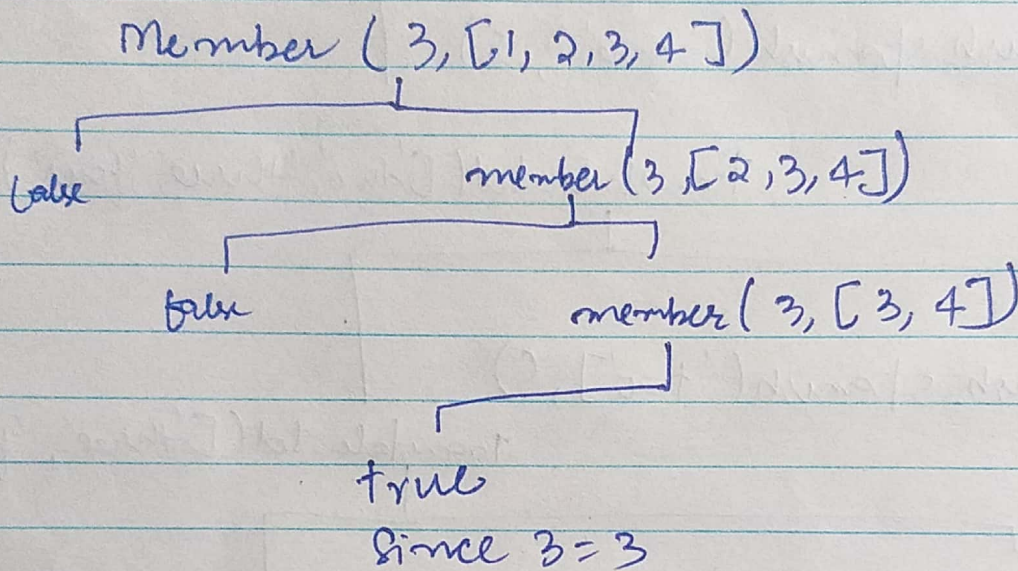
Draw Proof tree for Query :
? - translate_td (["One", "Two", "three", 'four'], S).
? - translate_td (E, ['Uno', 'seis', 'Ocho', 'Nueue', 'Diez']).

translate_td ([[English | English list], [Spanish |
    Spanish list]) :-

english_Spanish (english, spanish), translate_td (
    english list, spanish list).

these Rules are Recursion
∴ the recursive tree will be as follows

translate_td (['one', 'two', 'three', 'four'], S)

english_spanish ('one'), S)

Translate_td (['two', 'three', 'four']), S).

english_spanish ('two'], S)

Translate_td (['three', 'four'], S

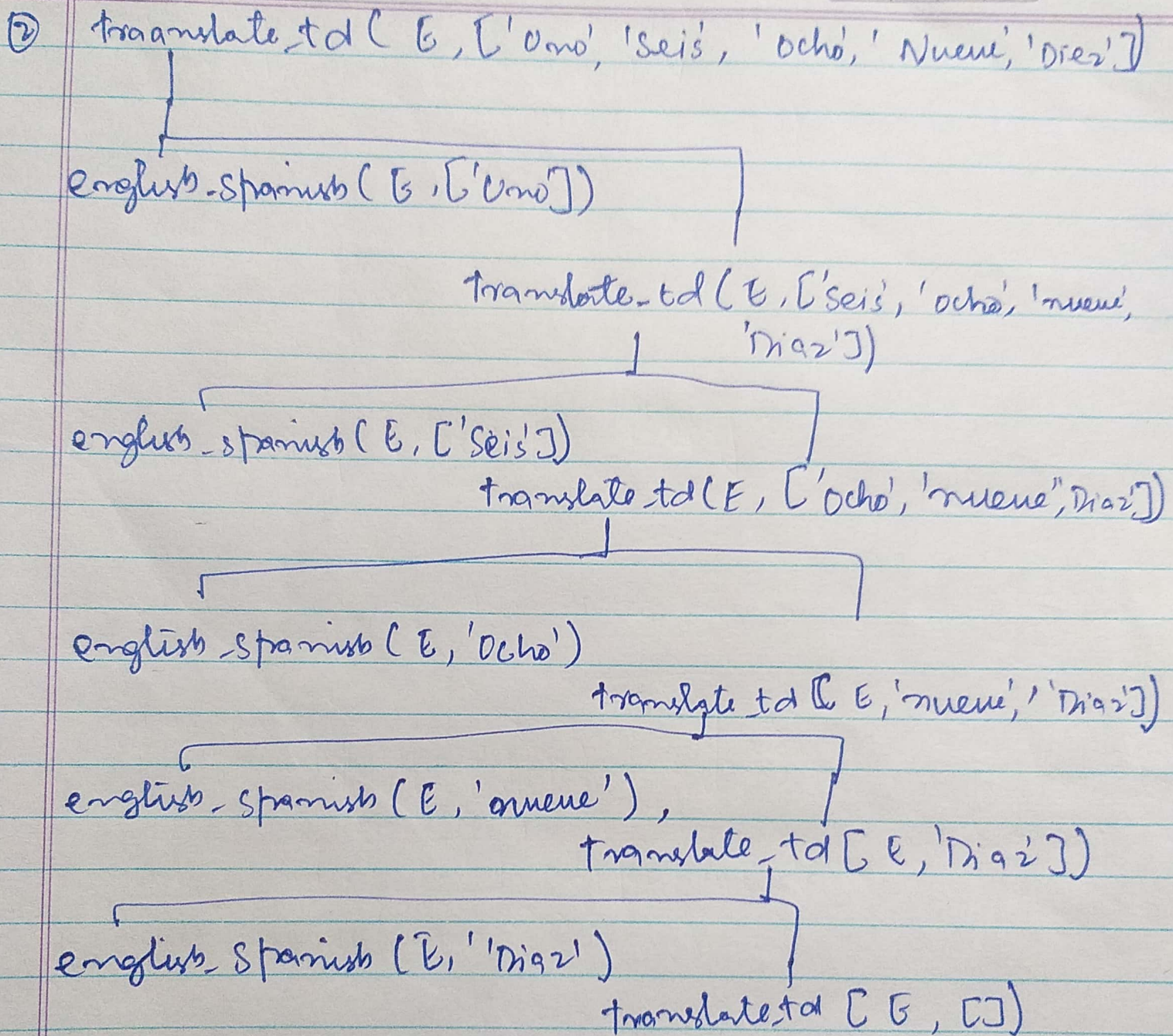english_spanish ('three'], s)

translate_td (['four'], S)

english_spanish ('four'), S)

translate_td ([ ], S).

So, S list can be seen as follows:

S [uno | Dos] tris | cuatro]

② translate_td (E , ['Uno', 'Seis', 'Ocho', ' Nueue', 'Diez']

english_spanish (E ,['Uno'])

translate_td (E , ['seis', 'ocho', 'nueue', 'Diaz'])

english_spanish (E , ['Seis'])

translate_td (E , ['ocho', 'nueue', 'Diaz'])

english_spanish (E , 'Ocho')

translate_td (E , 'nueue', 'Diaz'])

english_spanish (E , 'onueue') ,

translate_td [E , 'Diaz'])

english_spanish (E , 'Diaz')

translate_td (E , [])

Resulting E list is

| One | Six | eight | nine | ten |