UPDATES ASSIGNMENT 2

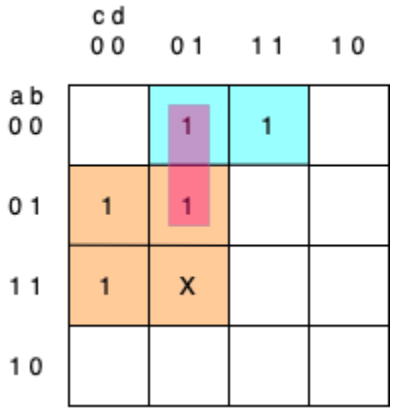
1. Please use the updated function template:

```
comb_function_expansion(func_TRUE, func_DC):
"""
   determines the maximum legal region for each term in the K-map function
        Arguments:
          func_TRUE: list containing the terms for which the output is '1'
          func_DC: list containing the terms for which the output is 'x'
        Return:
          a list of terms: expanded terms in form of Boolean literals
"""
```

2. Please use the updated sample cases:

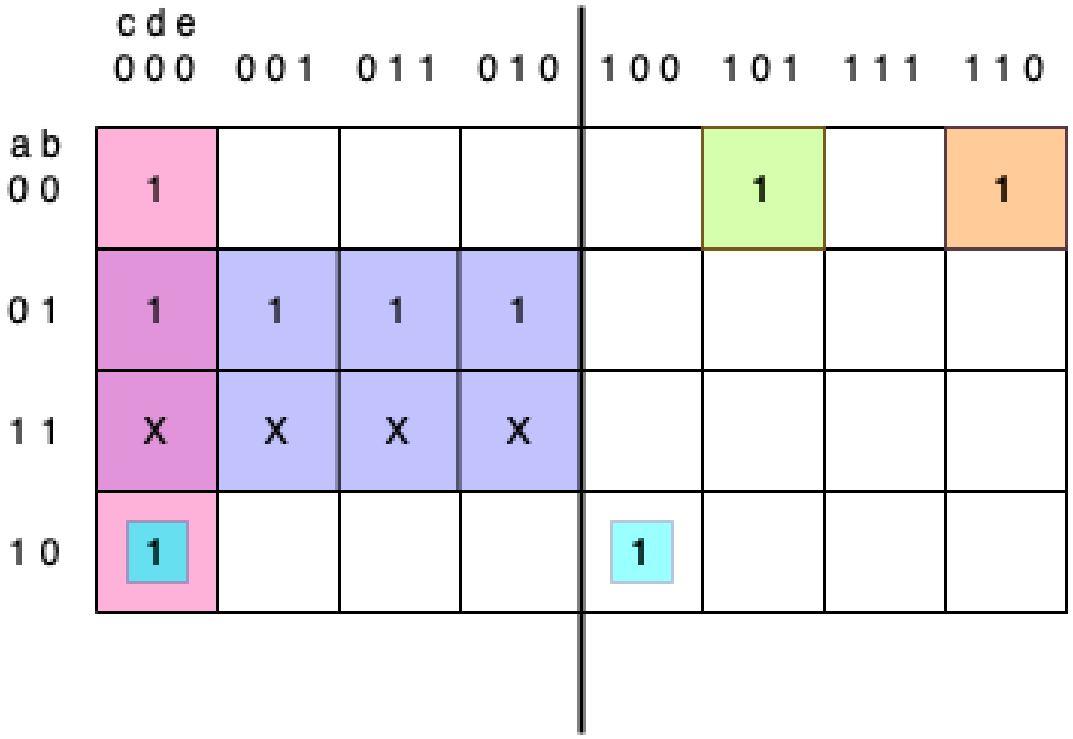
   a. Sample Case I

```
Input:
func_TRUE = ["a'bc'd'", "abc'd'", "a'b'c'd", "a'bc'd", "a'b'cd"]
func_DC = ["abc'd"]

Output:
["bc'", "bc'", "a'c'd", "bc'","a'b'd"
```



   b. Sample Case II

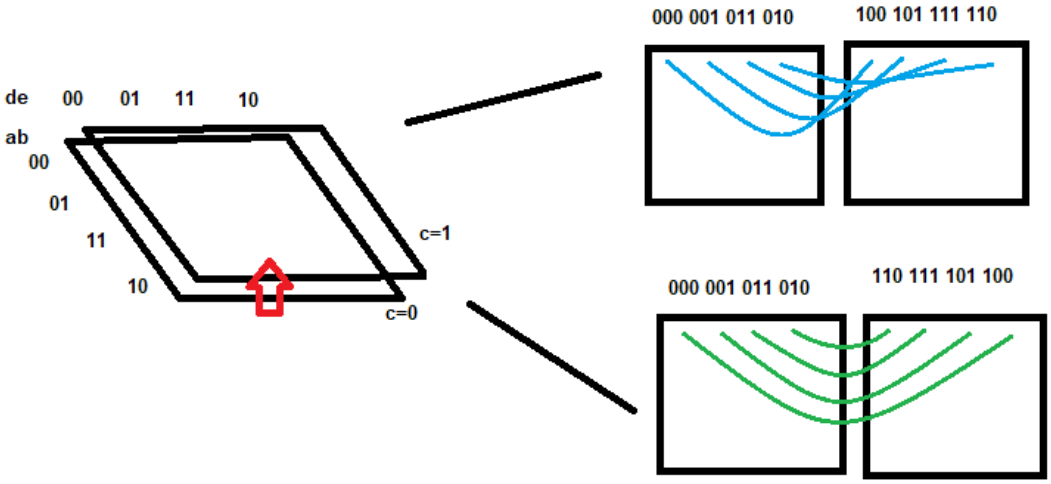
```
Input:
func_TRUE = ["a'b'c'd'e'", "a'b'cd'e", "a'b'cde'", "a'bc'd'e'",
             "a'bc'd'e", "a'bc'de", "a'bc'de'", "ab'c'd'e'", "ab'cd'e'"]
func_DC = ["abc'd'e'", "abc'd'e", "abc'de", "abc'de'"]

Output:
["c'd'e'", "a'b'c'd'e'", "a'b'cde'", "bc'", "bc'", "bc'", "bc'",
"c'd'e'", "ab'd'e'"]
```



3. The code developed should work on at least these inputs:
   - 2-input k-map this will 2*2
   - 3-input k-map this will 2*4
   - 4-input k-map this will 4*4
   - 5-input k-map this will 4*8
   - 6-input k-map this will 8*8
   - **and more**

   The literals within a term will be English alphabets only, and will be in increasing order within the input. N = 6 , will have [a,b,c,d,e,f] and not [a,f,r,t,v,x]

4. Grey Codes can be used while solving the assignment. Wrapping around the edges is allowed. When it comes to K-maps with five or more literals, it cannot be interpreted as planer. Refer to the diagram below



These are the legal ways to move across the cells on different layers

```
000 -> 100
001 -> 101
011 -> 111
010 -> 110
```

5. Consider a smaller example:

---	c'd'	c'd	cd	cd'
a'b'	1	1	0	0
a'b	1	1	0	0
ab	1	1	0	0
ab'	1	1	0	0

For each one of these cells, you need to find the maximum possible valid region. Let's go term by term:

```
a'b'c'd' ->  c'
a'b'c'd  ->  c'
a'bc'd'  ->  c'
a'bc'd   ->  c'
abc'd'   ->  c'
abc'd    ->  c'
ab'c'd'  ->  c'
ab'c'd   ->  c'
```

In the output, you should be returning all these 8 values. At each literal we have taken maximum possible region.

Additionally consider the K-Map below, `func_TRUE = ["a'b'", "a'b", "ab", "ab'"]`.

---	b'	b
a'	1	1
a	1	1

The output be [None, None, None, None] as it was in assignment 1. Because "1" In Karnaugh map does not mean a region while None will correspond to the whole map.

6. *Demo Instructions*

In the Demo you will be asked to print intermediate results, i.e. for a given term print which all terms it combines to form the resulting term after expansion. Instead of legal terms you can also print the next legal region used in the expansion. Find below an example for the same :

```
N = 3
Current term expansion: "a'b'"
Next Legal Terms for Expansion: "a'bc", "a'bc'"
Expanded Term: "a'"

----------------------------------------------------------------------

N = 4
Current term expansion: "bd"
Next Legal Terms for Expansion: "a'bc'd'", "a'bcd'", "abc'd'",
"abcd'"
Expanded Term: "b"

----------------------------------------------------------------------

N = 6
Current term expansion: "d'ef"
Next Legal Terms for Expansion: "a'b'c'def" , "a'bcdef" , "ab'cdef" ,
"abc'def","a'b'cdef" , "a'bc'def" , "ab'c'def" , "abcdef"
Expanded Term: "ef"

----------------------------------------------------------------------

N = 8
```

```
Current term expansion: "bc'e'gh"
Next Legal Terms for Expansion: "a'bc'defgh" , "a'bc'd'efgh" ,
"abc'd'efgh" , "abc'defgh" , "a'bc'def'gh" ,"a'bc'd'ef'gh" ,
"abc'd'ef'gh" , "abc'def'gh"
Expanded Term: "bc'g"

----------------------------------------------------------------------
```