COL215 Assignment 1 Report

In the assignment, we are asked to define a function, which takes kmap-function and a term as input and returns three-tuple: (top-left coordinate, bottom right coordinate, Boolean value) where the Boolean value indicates whether the region is legal or illegal.

Approach:

We decided to handle each type of kmap function separately (2-input, 3-input, 4-input kmap).

First, we transformed 2D kmap function list into 1D list named 'y'.

1. For 2-input kmap:
   We defined three empty sets x, x1 and x2.
   We updated the set x by adding the coordinates corresponding to the value of a (term[0]). 'If' statements have been used in both the cases, so that when the value of a is 'None', it should consider both the cases.
   Next, if the value of b (term[1]) is 0, we created a new set s1 containing the corresponding coordinates. We updated the set x1 by the intersection of set s1 and set x. Similar approach is taken when the value of b is 1 (set x2 is updated) and when b is 'None', both x1 and x2 are updated.
   New set x3 contains the required region corresponding to the given term which is union of sets x1 and x2.
   In order to get a sorted list of coordinates contained in the region, we created a new list h containing elements of set x3 and sorted it according to the first element of each coordinate.
   We traversed along the list 'h' and accessed the indices of elements in y, using the coordinates in list h, with the help of formula: index q = (h[i][0]*2)+h[i][1]
   Then, we checked the values in list y at the corresponding indices, if the value is 0, we returned a tuple containing the required coordinates and the boolean value 'False'.
   The case when the input term points to a single cell in kmap has been separately handled.

If the list y does not contain 0, for any value corresponding to coordinates in h, then we return a tuple containing the required coordinates and boolean value ‘True’.

2. For 3-input kmap:
   We defined five sets x, x1, x2, x3 and x4.
   We updated the set x, by adding the coordinates corresponding to the value of c (term[2]).
   Next, if the value of b (term[1]) is 0, we created a new set s1 containing the corresponding coordinates. We updated the set x1 by the intersection of set s1 and set x. Similar steps are followed when value of b is 1.
   We then take similar approach for the value of a (term[0]).
   We take union of x1 and x2 (=x5) and union of x3 and x4 (= x6). And then we define set x7 as the intersection of x5 and x6. We created a list ‘h’ of the elements of set x7. We first sorted h according to the first element of coordinates. Then in order to check the values at the coordinates, we traversed along the list ‘y’. And while traversing, we checked if the region is wrapping or non-wrapping by applying the condition whether the region contains the next consecutive row (consecutive to the first occurring row). And similarly, we return the boolean value False if the region contains 0, otherwise True.

3. For 4-input kmap:
   We first created 7 empty sets.
   We then updated the set x by the coordinates corresponding to the value of c (term[2]). We created set s1 containing the coordinates for which the value of b is 0. We updated the set x1 by the intersection of x and s1. Similar approach is followed for updating the sets x2, x3, x4, x5, x6.
   Finally, we get a set x11 containing the coordinates of the region corresponding to the given term. We then created two lists h and g1 out of the elements of set x11. We sorted the list h according to the first element of the coordinates whereas g1 according to the second element of coordinates. We used the list h1 for checking whether the region is wrapping or unwrapping column-wise and g1 for row-wise. By traversing

through the list y we checked if our corresponding region contains 0 or not and returned the corresponding tuple accordingly.

TEST CASES:

1.py > ...

```
k1=[[1,'x'],[0,1]]
t1=is_legal_region(k1,[None,None])
t2=is_legal_region(k1,[1,1])
t3=is_legal_region(k1,[0,None])
k2=[[1,'x',0,1],['x',1,0,'x']]
t4=is_legal_region(k2,[None,0,None])
t5=is_legal_region(k2,[None,None,None])
t6=is_legal_region(k2,[0,1,0])
k3=[[1,0,'x',0],[1,1,1,0],[1,'x','x',0],[1,0,1,0]]
t7=is_legal_region(k3,[None,0,None,0])
t8=is_legal_region(k3,[None,None,None,0])
t9=is_legal_region(k3,[1,1,0,0])
t10=is_legal_region(k3,[None,None,None,None])
```

2.py

```
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[0,0,0,0]) output = ((0,0),(0,0),False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[0,0,0,None]) output =((0, 0), (1, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[0,0,None,0]) output =((3, 0), (0, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[0,None,0,0]) output =((0, 0), (0, 1), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,0,0,0]) output =((0, 3), (0, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[1,0,None,1]) output =((1, 3), (2, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[1,0,None,0]) output =((3, 3), (0, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[1,1,1,1]) output =((2, 2), (2, 2), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[1,None,1,1]) output =((2, 2), (2, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,None,None,None]) output =((0, 0), (3, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,None,None,0]) output =((3, 0), (0, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,None,None,1]) output =((1, 0), (2, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,None,0,None]) output =((0, 0), (1, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,0,None,None]) output =((0, 3), (3, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[0,None,None,None]) output =((0, 0), (3, 1), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[0,None,None,1]) output =((1, 0), (2, 1), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[0,None,0,None]) output =((0, 0), (1, 1), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,0,None,0]) output =((3, 3), (0, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,0,1,None]) output =((2, 3), (3, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,0,0,None]) output =((0, 3), (1, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,0,None,1]) output =((1, 3), (2, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,1,None,0]) output =((3, 1), (0, 2), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[1,None,None,0]) output =((3, 2), (0, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[0,0,None,None]) output =((0, 0), (3, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[None,0,0,0]) output =((0, 3), (0, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[0,0,None,0]) output =((3, 0), (0, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]],[1,0,None,0]) output =((3, 3), (0, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0]], [0,0,0]) output =((0, 0), (0, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0]], [1,1,1]) output =((1, 2), (1, 2), True)
# is_legal_region([[0,1,1,0], ['x',1,'x',0]], [None,None,None]) output =((0, 0), (1, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0]], [None,None,0]) output =((0, 0), (0, 3), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0]], [None,0,None]) output =((0, 3), (1, 0), False)
# is_legal_region([[0,1,1,0], ['x',1,'x',0]], [0,None,None]) output =((0, 0), (1, 1), False)
# is_legal_region([[0,1], ['x',1]],[0,0]) output =((0, 0), (0, 0), False)
# is_legal_region([[0,1], ['x',1]],[1,1]) output =((1, 1), (1, 1), True)
# is_legal_region([[0,1], ['x',1]],[None,None]) output =((0, 0), (1, 1), False)
# is_legal_region([[0,1], ['x',1]],[0,1]) output =((1, 0), (1, 0), True)
# is_legal_region([[0,1], ['x',1]],[1,0]) output =((0, 1), (0, 1), True)
# is_legal_region([[0,1], ['x',1]],[0,None]) output =((0, 0), (1, 0), False)
# is_legal_region([[0,1], ['x',1]],[None,0]) output =((0, 0), (0, 1), False)
# is_legal_region([[0,1], ['x',1]],[None,1]) output =((1, 0), (1, 1), True)
# is_legal_region([[0,1], ['x',1]],[1,None]) output =((0, 1), (1, 1), True)
```

Made By: Disha Shiraskar (2021CS10578), Srushti Junghare (2021CS50613)