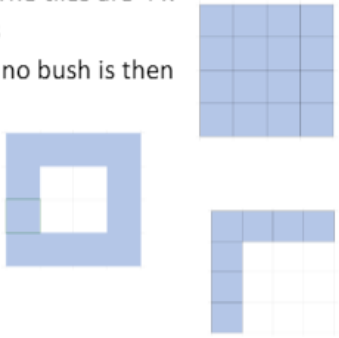
AI Project 2

Aditya Somani(G44043134)

Problem:

Given

- You are given a landscape on which certain "bushes" grow, marked by colors: 1, 2, 3, 4.
- The landscape is of square shape, so, it might be 100 x 100 or 200 x 200 etc.
- You are given a set of "tiles" which are of three different shapes. The tiles are 4 x 4. One tile only covers part of the landscape. Here are the shapes
    - Full Block: A "full block" tile covers the full 4 x 4 area, and no bush is then visible in that patch.
    - An "outer boundary" tile covers the outer boundary of the 4 x 4 area, and any bush in the middle part is visible.
    - An "EL" shaped tile covers only two sides of the area.
- You are given a "target" of which bushes should be visible after you have finished placing the tiles.



Solution:

Algorithm:

1. Read the input file and store the Landscape in a List, given tiles as Object and targets as Dictionary.
2. We consider only the EL shaped and Outer boundary tiles to get the required targets and cover rest of the landscape using Full blocks.
3. We place the EL shaped and Outer boundary tiles in each possible point (every 4th point in every 4th row) in the landscape and calculate the bushes visible in each and store it in a List for Backtracking.
4. To **find the MRV** we check the possible combinations of both EL shape tiles and Outer Boundary tiles according to combination of possible locations and available tiles.
5. We start **forward checking** on the MRV tile with the possible combinations and find the minimum of all the difference between the Target and Current bushes.
6. We use these steps to **filter the domain(AC-3)** and reduce domain search space by filtering any combination exceed bush target.
7. We sort these combinations in descending order (using the min () of difference between target and current) to **find the LCV**.
8. We update the target with the MRV tile and start forward checking with these updated targets and "not MRV" tile.
9. As soon as we find valid combination which satisfies the target, we return the current combination.
10. We use the "MRV tile" combination + "not MRV" tile combination and fill full blocks in rest of the spaces to get the result.

MRV Heuristic used- We check the possible combinations of EL shaped tiles and the outer boundary tile and find the minimum value between them. We assign the minimum value tile as the MRV tile. And find the possible combinations of the MRV tile which are satisfying the constraints (equal or below the target).

```
def getMrv(elCombinationCounts, outerCombinationCounts):
    mrv = None
    if elCombinationCounts > outerCombinationCounts:
        mrv = ['OUTER_BOUNDARY', 'EL_SHAPE']
    else:
        mrv = ['EL_SHAPE', 'OUTER_BOUNDARY']

    return mrv
```

LCV Heuristic used- We sort the combinations using target- current value in descending order to get the least constraint values in the start of the list.

```
def orderLcv(combinations):
    combinations.sort(key=lambda k: k[2], reverse=True)
```