Why1.pdf

1st Why : **Assuming that you number the tiles in the natural way, the tiles in the first tiling will run from 0 to 120, and the tiles in the second tiling will run from 121 to 241.**

1st Answer: Each tiling has the form 11x11 form and we look one by one for each tile. Since it has 11x11 form, first tiling will have 11x11 = 121 choices. It starts with 0 so it should be able to run until 120. Second tiling also have 121 choices, but it should start from 121 so it will end up at 241.

2nd Why: **For example, the point from the first example in the training set above, in1=0.1 and in2=0.1, or 0.1,0.1, will be in the first tile of the first seven tilings, that is, in tiles 0, 121, 242, 363, 484, 605, 726.**

2nd Answer: First of all, each tile has 0.6 length and 0.6 width. Since our in1 = 0.1 and in2 = 0.1, it will absolutely will be in the first tile. As long as it passes 0.6 it will not be in the first tile anymore. We change each index of width and length by i*0.6/(numTiling=8). Therefore in each iteration we will get close to 0.6 until 7th tiling. I will write down each iteration. I also mention that I do not repeat it for index2 because it will be same as index1;

(0.1 + 0 * 0.6/8) = 0.1 → first tiling (smaller than 0.6, so it is in the first tile)
(0.1 + 1 * 0.6/8) = 0.175 → second tiling (smaller than 0.6, so it is in the first tile)
(0.1 + 2 * 0.6/8) = 0.25 → third tiling (smaller than 0.6, so it is in the first tile)
(0.1 + 3 * 0.6/8) = 0.325 → fourth tiling (smaller than 0.6, so it is in the first tile)
(0.1 + 4 * 0.6/8) = 0.4 →fifth tiling (smaller than 0.6, so it is in the first tile)
(0.1 + 5 * 0.6/8) = 0.475 → sixth tiling (smaller than 0.6, so it is in the first tile)
(0.1 + 6 * 0.6/8) = 0.55 → seventh tiling (smaller than 0.6, so it is in the first tile)
(0.1 + 7 * 0.6/8) = 0.625 → eighth tiling (bigger than 0.6, so not in the first tile)

Therefore, we got for the first seven tilings as 0,121,242,363,484,726

3rd Why : **In the eighth tiling this point will be in the 13th tile (why?)**

3rd Answer: It will move with each iteration, but not only width but also length will move in a symmetric way since both are 0.1. Therefore, in the eighth tiling it will move from first tile to top right corner of it which is 13th tile.

4th Why: **which is tile 859**

4th Answer: the seventh tiling is 726, if it stays in the first tile for the eight tilings it should be 726 + 121 = 847. However, since it moved to 13th tile it should be 847 + 12 + 859 (I counted first tile as a zero index, therefore I added 12. If we count from 0 to 12 , we will have 13 tiles).

5th Why: **f you call tilecode(0.1,0.1,tileIndices), then afterwards tileIndices will contain exactly these eight tile indices. The largest possible tile index is 967**

5th Answer: We have 11x11= 121 grids. Let say the first eight tilings will be in the first tile, so we get 121 x 7 = 847 for this. However our eighth tiling can be at maximum 121st tile. Let say, if 8th tilings will be at 121st tile, then we will have 847+ 120 = 967 (As I did in the 4th answer, I counted the first tile as index 0)

6th Why: **Finally, the second and fourth examples should produce very similar sets of indices (they should have many tiles in common)**

6th Answer: Producing different sets of indices is the result of in1 and in2 that are given as functions parameters in the beginning.

```
printTileCoderIndices(4.0,2.0)
printTileCoderIndices(4.0,2.1)
```

In this called functions, the given parameter in1 is same and in2 is really close for both called function. Also, the difference for in2 is 0.1 which is pretty smaller than the size of one tile (0.6). Therefore, produced sets of indices are very similar, which are;

```
Tile indices for input ( 4.0 , 2.0 ) are :  [39, 160, 281, 403, 524,
645, 777, 898]
Tile indices for input ( 4.0 , 2.1 ) are :  [39, 160, 281, 403, 535,
656, 777, 898]
```