1) Answers to questions about part 1

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 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 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 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 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 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) and 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]

2) Explanation of part 2

After only 20 examples, my learned function will not yet look like the target function because the number of example is not enough to get better result since it cannot visit every tile. After I plot the graph for f20, most of the place was looks like flat. However, there are three peaks, but one of them are not high and there are seven valleys. I cannot give you exact number of the highest or width but I can say that, being high or deep means that those tiles are visited a lot more than the flat tiles. I said there is one hill which is not high because those tiles that the hill placed were not visited a lot compare to other hills. This rule applicable for the valley as well. The width depends on the how many distinct tiles are visited.

Instead of 11x11 tiles, using 11x21 tiles will give more accurate plot since there will be more grid. The number of visiting in one tile will be less, up to its value. We can think it as pixels, if there are more pixels we will have more accurate and beautiful result in a photo. Same thing happens to this 11x21 plot as well.

3) Answers to questions about part 2

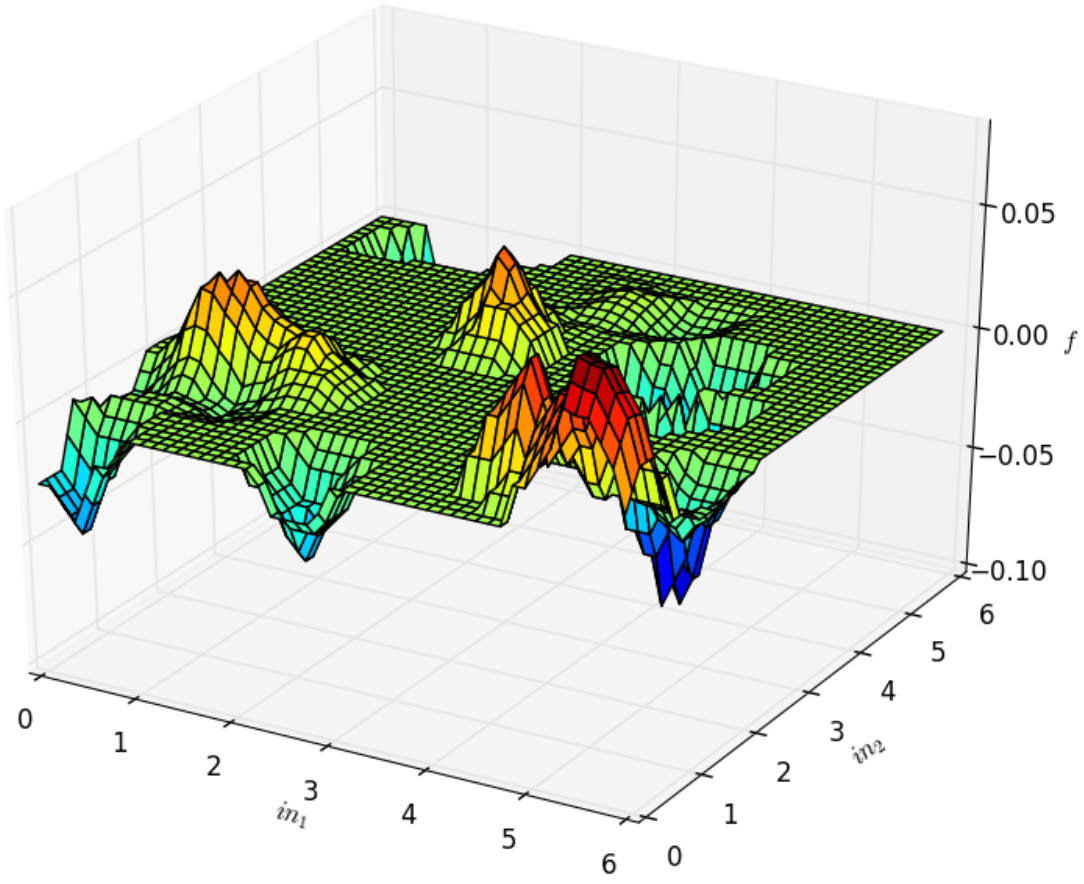
1st Answer: The second point (4.0, 2.0 , -1) and fourth point (4.0, 2.1, -1) are really close to each other. As I mentioned in the why1 file , there are 6 common tilings for both of them. Since the program learned those 6 common tilings in the second point, it is expected to being nonzero for the fourth point.

2nd Answer: We first get 0.25 because the test2() function calculates MSE(10000) without training function, which call learn function that I wrote as well. However, for other MSE(10000) function, which is in the for loop the error gets smaller and smaller, but never become 0. Because the learn function gets close to target function, however, the targetFunction function has normal(0,0.1) in it. This normal distribution never considered in the learned function, therefore the

4) MSEs

The estimated MSE: 0.247899203919
The estimated MSE: 0.0548548291414
The estimated MSE: 0.0201174817201
The estimated MSE: 0.0139721410702
The estimated MSE: 0.0127727136547
The estimated MSE: 0.0119166910642
The estimated MSE: 0.0114390370367
The estimated MSE: 0.011506034811
The estimated MSE: 0.0110168043967
The estimated MSE: 0.0112275339001
The estimated MSE: 0.0111551977283

5) F2



6) F10000