Running with 1 in the corner successfully worked. I began by changing the array to numerical order, then added a function for row 1 which checked 4 places so it would continue if the numbers are == to 64 (65-1) and after running for 36 minutes, got the number 70944652.

Similar to this, I added the 1 to the second spot, so the top row was numbered 1 0 2 3 4 and had to revert the rows and cols to follow this pattern. In a little over 40 minutes I got 101264196 different squares.

Following this same pattern, I moved the 0 index (which contained the 1) so my top row looked like - 12034. After running it once, I got over 100million different squares, and after reviewing my code, found a stupid mistake. 36 minutes later, I got 98730584 squares. Success.

I am still short 5 million squares from the official count. Finally, adding 1 to the center of the square took a little bit of work. I moved the first index to the center, replacing the 6th index. This left the top row 01234 and the 3rd row 21 12 6 14 22. I did this pretty quickly as I was running out the door and got a little over 26 million, which got me around 297 million different squares. So I looked through and realized I missed something big. I re-rearranged the square so the 0 index pushed the others over to look like this

1, 2, 3, 4, 19,

5, 17, 9, 20, 21,

12, 6, 0, 14, 22,

23, 10, 18, 7, 24,

11, 13, 16, 15, 8

a[0] = 1

There must be a better way...

I decided to change the original ordering of the array

/\*Indices

0, 1, 2, 3, 4,

18, 5, 16, 8, 18,

20, 11, +, 13, 20,

22, 9, 17, 6, 23,

10, 12, 15, 14, 7

\*/

int a[24] = {2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25};

At this point I’m willing to try anything and after 30 seconds, received 1.5million different ways. I’m throwing in the white flag. ~272 is where I ended.