For my work I coded different functions for each of the transposes. I checked the value of both M and N and then from whatever value they were, it would do the appropriate transpose.

32x32:

For this matrix I used the video in my credits to help guide me to code a general matrix transpose. I loop through using the block size of 8, since 8 was the most appropriate I found for the transpose of a 32x32 matrix, the cols(col) and rows(row) of B and then set them equal to the rows(i) and cols(j) of A since we can't alter the A array. If the values for i and j were the same then I would use a temporary variable to hold the value of A[i][j] and another temp variable to hold the value of i. Then I would check if the values of row and col were the same and if they were then I would assign the value of B at indices of the variable holding A[i][j[and assign it to the 2nd var.

61x67:

The CS tutors helped me with this function. They said the logic is going to be almost the same as the 32x32, the only thing that would be a big difference is the block size. This time the block size would be 16 instead of 8. Also in my for loop instead of i<row+blockSize(like in my 32x32 transpose) I had to check 2 things. First was to make sure that i was less than N and j was less than M, and to check if i<row+blockSize and j<col+blockSize. I tried it out and when I finished the correctness was 1 and my misses were under 2000 so I left that as my conclusion.

64x64:

This by far was the hardest one to make sure to get under 1300 misses. This is where I found the video https://www.youtube.com/watch?v=huz6hJPI_cU. This video helped me understand how to go about being cache friendly when doing my work. The tutors had helped me with understanding how to reduce the amount of cache misses by dividing the matrix into smaller blocks. This helped by reducing the number of cache lines that needed to be evicted from the cache. I start with 8 sized blocks and assign temporary variables to various rows and cols of A and set them to indexes in matrix B. Then I move to block sizes of 4 and repeating what I did with a size of 8. One quick thing I found while coding this part is the way I initialized temp variables affected the amount of misses.