1. [3 points] We have seen many issues that can affect the performance of a kernel. State three issues that can affect the performance of a kernel, in no more than one sentence each.

•

•

•

2. [3 points] State three useful usages of streams.

•

•

•
3. [4 points] Assume we have the following kernel for matrix multiplication. We launch it with matrices of size 1000x1000 and each block is 16x16 threads (i.e. 2D blocks of 16x16). How many warps will have control divergence? Justify

```
__global__ void MatrixMulKernel(float* M, float* N, float* P, int Width)
{
    int Row = blockIdx.y*blockDim.y+threadIdx.y;
    int Col = blockIdx.x*blockDim.x+threadIdx.x;
    if ((Row < Width) && (Col < Width)) {
        float Pvalue = 0;
        for (int k = 0; k < Width; ++k) {
            Pvalue += M[Row*Width+k] * [k*Width+Col];
        }
        P[Row*Width+Col] = Pvalue;
    }
}
```

4. [3 points] For a vector addition, assume that the vector length is 4000, each thread calculates one output element, and the thread block size is 512 threads. The programmer configures the kernel launch to have a minimal number of blocks to cover all output elements. How many threads will be in the grid? To get full credit, show all steps.
5. [2 points] In class, we have seen tiling as a useful technique in matrix multiplication. State two scenarios where tiling is useful at:

•

•

6. [2 points] We know that CUDA does not allow synchronization among threads in different blocks. Suppose CUDA allows this. State one potential problem that may arise.

7. [2 points] We have seen many type of memories in the GPU. One of them is the local memory. Given that registers are used on per thread basis, what is the point of having a local memory?

8. For the following sequential code (Assume matrix B is initialized to 0):

```c
#define N 64
float A[N][N];
float B[N][N];
for (k=0; k<N; k++) {
    for (j=0; j<N; j++) {
        for (i=0; i<N; i++) {
            B[i][j] += A[j][k];
        }
    }
}
```
Assume the GPU we will use has 8 SMs, and can have at most 8 blocks per SM, 512 threads per block, and 1024 threads per SM. Each SM has 64KB of shared memory. The GPU also has 6GB of global memory and 64KB of constant memory.

a. [2 points] If you write a CUDA version of that code, what is the best location to store the matrix A? Why?

b. [3 points] What will be the geometry of the CUDA kernel (grid and block sizes and dimensions) to get the most parallelism? Justify your choice.


d. [4 points] Assume there no shared memory in the SM. Write the kernel that will be executed by each thread depending on your answer of question b above. We assume matrices A and B are already at the device global memory.