HW4 GPU Compute CUDA Questions:

Please provide a written explanation along with answer selection to receive full credit.

1. For the tiled matrix-matrix multiplication kernel discussed in Lectures 3, 4 & 5, if we use a 32X32 tile, what is the reduction of memory bandwidth usage for input matrices A and B?

    a. 1/8 of the original usage
    b. 1/16 of the original usage
    c. 1/32 of the original usage
    d. 1/64 of the original usage

2. For the tiled single-precision matrix multiplication kernel as discussed in Lectures 3, 4 &5, assume that the tile size is 32X32 and the system has a DRAM burst size of 128 bytes. How many DRAM bursts will be delivered to the processor as a result of loading one A-matrix tile by a thread block?

    a. 16
    b. 32
    c. 64
    d. 128

3. Assume a tiled matrix multiplication that handles boundary conditions only on the right edge and lower (bottom) edges. Assume that we use 32X32 tiles to process square matrices of 1,000x1,000. Within EACH thread block, what is the maximal number of warps that will have control divergence due to handling boundary conditions for loading input matrix M (see L4 slides 31-35) tiles throughout the kernel execution?

    a. 32
    b. 24
    c. 16
    d. 8

4. We are to process a 600X800 (800 pixels in the x or horizontal direction, 600 pixels in the y or vertical direction) picture with the PictureKernel(). That is m's value is 600 and n's value is 800.

```
__global__ void PictureKernel(float* d_Pin, float* d_Pout, int n, int m) {
    // Calculate the row # of the d_Pin and d_Pout element to process
    int Row = blockIdx.y*blockDim.y + threadIdx.y;
    // Calculate the column # of the d_Pin and d_Pout element to process
    int Col = blockIdx.x*blockDim.x + threadIdx.x;
    // each thread computes one element of d_Pout if in range
    if ((Row < m) && (Col < n)) {
        d_Pout[Row*n+Col] = 2*d_Pin[Row*n+Col];
    }
}
```

Assume that we decided to use a grid of 16X16 blocks. That is, each block is organized as a 2D 16X16 array of threads. How many warps will be generated during the execution of the kernel?

(A) 37*16
(B) 38*50
(C) 38*8*50
(D) 38*50*2

5. In Question 14, how many warps will have control divergence?

a. 37 + 50*8
b. 38*16
c. 50
d. 0

6. In Question 14, if we are to process an 800x600 picture (600 pixels in the x or horizontal direction and 800 pixels in the y or vertical direction) picture, how many warps will have control divergence?

e. 37+50*8
f. 38*16
g. 50*8
h. 0

7. In Question 14, if are to process a 799x600 picture (600 pixels in the x direction and 799 pixels in the y direction), how many warps will have control divergence?

i. 37+50*8
j. (37+50)*8
k. 50*8
l. 0

8. Assume the following simple matrix multiplication kernel

```
__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] * N[k*Width+Col];}
        P[Row*Width+Col] = Pvalue;
    }
}
```

If we launch the kernel with a block size of 16X16 on a 1000x1000 matrix, how many warps will have control divergence?

m. 1,000
n. 500
o. 1,008
p. 508

9. If a CUDA device's SM (streaming multiprocessor) can take up to 1,536 threads and up to 8 thread blocks. Which of the following block configuration would result in the largest number of threads in each SM?

q. 64 threads per block
r. 128 threads per block
s. 512 threads per block
t. 1,024 threads per block

10. Assume that we want to use each thread to calculate two (adjacent) output elements of a vector addition. Assume that variable i should be initialized with the index for the first element to be processed by a thread. Which of the following should be used for such initialization to allow correct, coalesced memory accesses to these first elements in the following statement?

if(i<n) C[i] = A[i] + B[i];

(A) int i=(blockIdx.x*blockDim.x)*2 + threadIdx;
(B) int i=(blockIdx.x*blockDim.x + threadIdx.x)*2;
(C) int i=(threadIdx.x*blockDim.x)*2 + blockIdx.x;
(D) int i=(threadIdx.x*blockDim.x + blockIdx.x)*2;

11. Continuing from Question 20, what would be the correct statement for each thread to process the second element?

    (A) If (i<n) C[i+1] = A[i+1] + B[i+1];
    (B) If (i+1<n) C[i+1] = A[i+1] + B[i+1];
    (C) If (i+threadIdx.x < n) C[i+threadIdx.x] = A[i+threadIdx.x] + B[i+threadIdx.x];
    (D) if(i+blockDim.x < n) C[i+blockDim.x] = A[i+blockDim.x] + B[i+blockDim.x];

12. Assume the following simple matrix multiplication kernel

```
__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] * N[k*Width+Col];}
     P[Row*Width+Col] = Pvalue;
   }
 }
```

    Based on the coalesced access judging criterion in Lecture 5 (slide 35), which of the following is true?

    (A) M[Row*Width+k] and N[k*Width+Col] are coalesced but P[Row*Width+Col] is not
    (B) M[Row*Width+k], N[k*Width+Col] and P[Row*Width+Col] are all coalesced
    (C) M[Row*Width+k] is not coalesced but N[k*Width+Col] and P[Row*Width+Col] both are
    (D) M[Row*Width+k] is coalesced but N[k*Width+Col] andt P[Row*Width+Col] are not