Instructions: Please submit your answers on Blackboard in any readily readable format (PDF, DOCX, images, etc).

1. Consider the following matrix multiplication code from the lectures:

```c
__global__ void kernelMatrixMult(float* C, float* A, float* B, size_t M, size_t N)
{
    size_t i = blockIdx.x * blockDim.x + threadIdx.x; //calculate the i (row) index
    size_t j = blockIdx.y * blockDim.y + threadIdx.y; //calculate the j (column) index
    if(i >= M || j >= M) return; //return if (i,j) is outside the matrix
    for(size_t n = 0; n < N; n++)  //for each element in the dot product
}
```

a) Which (if any) of the global memory instructions will result in coalesced memory transactions?

b) A GeForce Titan X has a maximum memory bandwidth of 336 GB/s and 6144 GFLOPS of 32-bit processing power. What percent utilization would you expect from the above kernel for large N?

c) How much would your utilization be increased if you wrote directly to global memory?

```c
float c = 0;
for(size_t n = 0; n < N; n++)  //for each element in the dot product
    c += A[n*M+i] * B[j*N+n]; //perform a multiply-add
```

2. How would you declare the following structure so that it is accessible on the device and easy to coalesce?

```c
struct color{
    float r;
    float g;
    float b;
};
```

How many unused bytes will be transferred during a coalesced transaction?

3. What sizes of device memory transactions are available on the Pascal architecture?

4. What sizes of global memory instructions are available on the Pascal architecture?

5. How would you declare a 64-bit read-only float that is accessible to all kernels?