GPU programming using CUDA

Q1 [Understanding 2-D blocks]
Modify the **pargpu.cu** code (from the class examples) where number of threads are specified as a two dimensional variable. In CUDA, **dim3** data type can be used to specify two or three dimensional elements. Specifically, number of threads in each dimension of a 2-D block should be parameters to the program. Your program should be executed as follows,

$ ./q1 {number of elements} {rows} {cols}

Where rows * cols <= 1024 (max threads per block}

Q2 [XOR based Checksum]
Given randomly generated N numbers {X_1, X_2, …..X_N} as input, find out the XOR sum as follows,

$$SUM = X_1 \text{ XOR } X_2 \text{ XOR } X_3 \text{ ….. XOR } X_N$$

You can use maximum O(1) extra space to perform the operations on GPU. Further, the input copied to the device memory is not required to maintain the old values after completion of the program.

Your program should take one argument, i.e., *number of elements* as command line parameter. In your program (main function) should generate random inputs before invoking the GPU kernel (refer class examples). *Submit only the CUDA source file in moodle*.

Submission format

Your submission should be a single archive (tar, zip etc.) named after your roll no. (<roll_no.zip>) which expands to a folder containing two bash source files (qn1.cu and qn2.cu).