**Discussion questions for OptiML: An Implicitly Parallel Domain-Specific Language for Machine Learning**

**Warm-ups**
1. Domain: machine learning problems/algorithms  
2. Goals:  
   a. Make parallelization easy for the programmer for some common machine learning problems  
   b. Remove the burden of thinking about optimization and hardware from the programmer  
      - program readability & productivity  
   c. Platform/hardware independent  
3. Challenges:  
   a. Programs are usually single-threaded unless the programmer manually optimizes it  
   b. Can only write for one architecture: either the CPU or GPU  
   c. Difficult to distill the actual algorithm (which is usually quite simple in pseudo-code) from code  
   d. Programmer time + run time is high

**Design Evaluation**
1. Abstractions and features  
   a. Data structures for matrices, vectors, graphs  
   b. D.S. collection ops: map, count, filter  
   c. Some built-in linear algebra  
   d. Parallelization completely abstracted away  
   e. Best-effort optimization  
2. Section 2.3 code  
   a. Implements k-means algorithm using the unfilconverged control structure  
   b. Uses matrix construction, vector construction, and map row op

3. Portability:  
   a. Compiler generates Scala, C++, and CUDA code and is completely cross-platform

4. Highly efficient code:  
   a. Generates an intermediate representation of code, which is then optimized  
      i. standard static optimizations  
      ii. static domain-specific optimizations  
   b. OptiML has restricted semantics and data structures  
   c. Runtime knows certain patterns of computation and can optimize for them  
      i. Best-effort computing as needed - trade off accuracy for performance

5. Implementation:  
   a. Embedded in Scala, uses the Delite runtime  
   b. Generates an intermediate representation of the code, which is optimized (“lightweight staging”)
c. Outputs Scala, C++, and CUDA code
6. Evaluation:
   a. Compared runtime to C++ and Matlab for different numbers of cores and with and without optimizations
   b. Compared code length and readability

Evaluating OptiML as a Domain-specific Language
1. Advantages/Disadvantages of embedding OptiML in Scala
   a. Advantages
      i. Scala is more widely known in industry than Haskell
      ii. Utilize all of Scala’s features
      iii. Good support for EDSLs in Scala
      iv. Interoperability with Java (JVM)
      v. Widely accessible
      vi. Scala Syntax
   b. Disadvantages
      i. Scala syntax
      ii. No type annotations?
2. Type system
   a. Only supports vector, matrix, graph, and scalar data structures
   b. Restriction of types enables optimizations
3. Runtime
   a. Based on Delite runtime
   b. Runtime also handles scheduling of operations on the underlying hardware and communication with the kernel
4. Libraries
   a. Pre-implemented ML algorithms
   b. Cross-validation libraries
5. Tools
   a. More error catch support: catch Scala errors and emit OptiML errors
   b. Debugger
   c. MapReduce interface?
   d. OptiML-specific visualization support
6. Is it a DSL?
   a. Yes! It’s got a specific domain (machine learning) and all features and omissions support the one goal of optimizing ML problems via implicit parallelization
7. Did it achieve its goals?
   a. Yes…
8. Improvement
   a. Type annotations
   b. API to add optimization pass to compiler