CS 442 Assignment 2:
Constructing a Simple Vector DSL using LMS & Delite

Due: Thursday, April 28th

1. Follow the instructions in the getting started guide at http://stanford-ppl.github.com/Delite to set up the project. For this class you should use the "stanford-cs442" git branch for both the Delite and LMS repositories. In the class branch you should find the assignment2 SimpleVector DSL in the DSLs folder, and the beginnings of an application using this DSL, ppl.apps.assignment2.SimpleVectorApp, in the apps folder. While following the instructions in the getting started guide, keep in mind that you'll need to modify the example classpath in order to include SimpleVector (not OptiML) and include the apps (framework and LMS remain the same).

2. Compile and generate SimpleVectorApp. A folder named "generated" should appear which contains the data structures and kernels generated for this application. Open up each of the Scala kernel files and attempt to match each kernel with the corresponding application source. Now launch the runtime with this application and make sure the printed result is what you expect.

3. Open up the two main files for the DSL: SimpleVector.scala, which defines the IR nodes and code generators available in the SimpleVector DSL, and Vector.scala, which defines all the operations on the Vector data type (the sole DSL data type in SimpleVector). Get a feel for what everything is doing.

4. Uncomment the commented line of scalar multiplication in SimpleVectorApp and try to compile again. What's wrong? Since scalar multiplication is commutative, it seems reasonable to represent both scalar-times-vector and vector-times-scalar operations with the same IR node. Therefore to add this functionality, we only need to enrich the available DSL syntax. Modify the DSL so that the app compiles again. If you generate this new program does the added line appear in the generated kernels. Why or why not?

5. A classic vector operation is the AXPY (aX+Y) function. Instead of making users call such a function explicitly, you want to them to be able to simply write something like "a*X+Y" and still be able to capture this operation in your IR.
    a. Add an AXPY IR node to the DSL by extending an appropriate Delite Op
    b. Optimize the IR by detecting when the AXPY pattern exists (based on scalar-multiplication and vector-addition nodes) and then use the AXPY node instead
    c. Modify the application and check the generated code to verify that the application uses an AXPY kernel

6. So far all of the arithmetic operations in our SimpleVector DSL returns a new Vector object, but sometimes we want an operation that mutates an existing vector. Add a "+=" operation to the DSL which does not allocate a new Vector but instead writes its result into the addend on the left hand side of the expression. Remember that the LMS Effects system requires the DSL to reflect that a given data structure is mutable and also reflect each write into that data structure.

7. Write an application for solving (a vector of) *N* first-order ODEs. The solver algorithm can be as simple or complicated as you want (a basic Euler method is sufficient). As you develop the application, think about how you can enhance the DSL to make developing this application (and hopefully others) more productive and the execution more efficient. You are free to modify the DSL any way you want. Add some comments in your code where appropriate.

8. Package your new DSL and application (just the files you added/modified) using your favorite file archive/compression tool (preferably one that your TAs have heard of before) and email it to the staff mailing list.