# Tree-based Expression Evaluator

# Programming Assignment #4

**Overview.** In Assignment 3, you implemented the expression evaluator using a stack. In this assignment, you will implement the same expression evaluator using a tree. To design and implement this version of the expression evaluator, you will use the Builder Pattern to create the tree, which is a Composite object. You will then use the Visitor pattern evaluate the expression tree.

Similar to Assignment 3, you must implement a fully functional program. The program should prompt the user for an expression, and then evaluate the expression using the design discussed above. Finally, the result should be printed to STDOUT.

**Runtime Requirements.** You are to create a complete program. The expression evaluator must be able to handle the following operators/tokens: +, -, /, *, %, (, ), integers (positive and negative). All expression will have a space between each token to simplify parsing. All input should come from the STDIN. The program must loop until the user types QUIT—notice the all caps. All output should go to STDOUT. Failure to follow the required input/output method will result in an automatic 25-point deduction. All assignments must run on Pegasus.

**Assignment files.** No files will be provided for this assignment. You are to update your MPC file accordingly so that you can build your expression evaluator. The name of the MPC file must be **assignment4.mpc**. It is highly recommended that you do not place all your classes in a single file and take advantage of modularization.

**Submissions.** All submissions must be placed on IU Github by the deadline. The Github repository must be **private** under your account and named **cs363-fall2018-assignment4**. Failure to upload all your source code and project files to a Github repo with this name will receive an automatic zero (0). You must add the TA and me (hilljh) as collaborators to the project. Failure to add the TA or me as a collaborator will result in an automatic zero (0).