Visitor Pattern

Author: Brian Crites

You <u>must</u> work in a group of two for this lab

The visitor pattern is used to visit multiple pieces of data, and then do some analysis on that data. Often this is done with the use of an iterator, which when paired with a visitor allows us to move over some data in a pre-defined way, visiting every piece. This also means that our analysis is decoupled from our iteration, which lets us reuse both analyses and iteration.

In this lab you will create a visitor class that collects information on each node it visits for printing. You will create a PrintVisitor class that has the following definition so that it fully integrates with the composite class we've already designed.

```
class PrintVisitor : public Visitor {
      private:
            std::string output;

      public:
            PrintVisitor();

            void rootNode();        #For visiting a root node (do nothing)
            void sqrNode();         #For visiting a square node
            void multNode();        #For visiting a multiple node
            void subNode();         #For visiting a subtraction node
            void addNode();         #For visiting an add node
            void opNode(Op* op);    #For visiting a leaf node

            void execute();         #Prints all visited nodes
};
```

You will also need to modify several of the composite classes to allow for the PrintVisitor to access them. You will add the function `void accept(Visitor*)` to any necessary composite classes, which will then call one of the functions above.

When you run this visitor along with a PreOrderIterator, after the iteration is complete you can call the execute() function on the visitor and it will print each node that it's visited.