Ahmad Hazimeh EECE 437 3/2/2017
201500700 Code Review

Code Heuristics:

• Lines of code:	752 (including whitespace)
• Comments to code:	0.001
• Number of classes:	24
• Number of files:	2
• Maximum level of inheritance:	3
• Code readability:	6 / 10
o No header-code separation, no comments	
o Simple methods and method calls	
• Style consistency:	8 / 10
o virtual and const specifiers used where suitable	
o Sibling classes (TAPlus, TAMinus…) followed the same style	

Code Review:

On Form:

- The name of the project is EECE_437_PA1, which indicates that it is the first program assignment in the EECE 437 course.
- The checked in items only include source and configuration files, which is preferable for compactness and distribution.
- The class declarations and their corresponding member function definitions are not separated, but are all included in one TAElement.h file.
  While this does impact readability, it makes searching limited to only one file, instead of keeping track of multiple open files or tabs, which might sometimes be more productive.
- All type names follow the naming convention: TA*Expression*
  All variable names follow the camelCase convention
  Method names don't follow any naming convention

On Operation:

- The code compiles fine with the provided examples, but it doesn't produce compilation errors where otherwise expected (TAPlus of TAInt and TADouble)
- The code runs and produces correct arithmetic results.
- The following is a list of unsatisfied and unreported operational requirements:
  - The code lacks support for variable names
  - The list() method prints invalid results.
    It prints the values of variables instead of their name.
  - The code lacks support for TAConstant and TAArrayAccess
  - TAArray class doesn't provide a size() member function

On Content:

The hierarchy proposed by the author was as follows:

- A TAElement is the highest level in the inheritance tree
- A TAElement can be list()ed and evaluate()d
- A TAElement can be operate()d on by defining what each supported EOperation should return
- TAType is an empty child of TAElement
- TABool, TAInt, and TADouble are direct children of TAType with a single state
- operate()ing on TABool, TAInt, and TADouble is well-defined
- TACompound is a child of TAType with multiple states
- TAPair is a TACompound with only 2 states
- TAArray is a TACompound with N states
- operate()ing on a TACompound recurses into operating on its individual states
- TAOperator is a direct child of TAElement
- operate()ing on a TAOperator recurses into operating on its evaluate()d TAElement
- evaluate()ing a TAOperator is operator-specific, and calls operate() of its operands, with the specified EOperation

The working design decisions in implementing this hierarchy seemed to rely on the Decorator design pattern, wherein certain TAElements (specifically, TAOperators) encapsulate one or two other TAElements, without having to specify the type chains at compile time (template pattern).

While it could have been more neatly implemented, the Decorator design pattern does seem to be able to accommodate the design requirements, given a more robust hierarchy.

However, compile-time type verification would not then be possible, and inconsistencies in types could only be reported at runtime through exceptions or return values.

Probing Further:

- Does the submission put the problem into context?
- Does the submission state principle design decisions?

The submission provides no additional documentation describing the architecture, the purpose, or the usage of the code.

The principal design decisions could be inferred from the hierarchy of the classes, as was described above.