RESOLVE Verifying Compiler Activities #2

Name: _____
Name: _____
One CM: _____

To Do:

- Navigate to the RESOLVE Compiler's Web IDE
- Click *Components* button, then from dialog choose *Programs*
- Do Activities 104b through at least 201
- Answer prompting questions below as you do the activities

Activity_104b_Reasoning_about_Max_and_If_Statements

Note: For this activity you are not allowed to fix *Mad_Max* by eliminating its first line of code

Activity: How did you fix *Mad_Max?* _____

_____

_____

Activity_104c_Reasoning_about_Design_by_Contract

Activity 2a: Why does *Do_Nothing_1* not verify after the *requires* clause has been eliminated?

_____

Activity 2b:

(a) If you remove the 1st assignment statement, *Do_Nothing_1* does not verify for 2 reasons, what are they?

_____

_____

(b) If you remove the 2nd assignment statement, *Do_Nothing_1* does not verify for 1 reason, what is it?

_____

Activity 3b: Uncomment *just* the call to Do_Nothing_1 for *Activity 3b*, why doesn't the call verify?

_____

Activity 3c: Now uncomment the *if* surrounding the call to *Do_Nothing_1*, and the call should verify. This *if* can be thought of as a *guard* to the call.

Activity 4a: Notice that the instructions say *change only the code* to make *Do_Nothing_2* verify.

(a) How did you get *Do_Nothing_2* to verify? _____

(b) How can *Do_Nothing_2* be made to verify by only *adding* code?

_____

(c) How can *Do_Nothing_2* be made to verify by changing the contract?

_____

Continued on back side

Activity_105_Reasoning_about_Loops_and_Recursion

With respect to operations *Clear_Num_0* and *Clear_Num_00:*

(a) What is the implicit *ensures* clause for both operations?_____

(b) There is a call to *Clear* in *Clear_Num_00*, using Java speak, what class must *Clear* come from?

_____

Questions about operation *Clear_Num_Recursively:*

(c) What is its *ensures* clause? _____

(d) What is its base case? _____

(e) At what point in its body does the verifier confirm the *decreasing* clause?

_____

(f) How many paths are there through its body? (Hint: envision a diagram of the code)

_____

Describe those paths:? _____

(g) At the very end of all those paths, what fact does the verifier know in order to confirm the ensures?

_____

(h) What is your response to: What if a client called *Clear_Num_Recursively* with a negative number?

_____

Questions about operation *Clear_Num_Iteratively:*

(i) What is its *ensures* clause? _____

(j) What is the loop exit condition? _____

(k) At what point in the loop body does the verifier confirm the *decreasing* clause?

_____

(l) At what two points does the verifier confirm the *maintaining* clause?

_____

(m) After the loop terminates, what facts does the verifier know in order to confirm the ensures?

_____

_____

Activity_201_Reasoning_about_Objects

Reference the *Grid_Positioning_Template* in the Web IDE's Components

(a) Fix the Confirm statements in *Main* so that *Main* verifies.

(b) Add and verify operation *Move_NE* by first *stubbing* it out, i.e., no contract and no code in the body.

Note: *Move_NE* will eventually be called by *Main* with the GP variable declared in Main's body

(c) Now add an implementation to *Move_NE*, you might need to add a contract.

(d) Add and verify a recursive implementation for *Move_to_Max_Right*

(e) Add and verify a recursive implementation for *Move_to_Max_Top*

(d) Add and verify an iterative implementation for *Move_to_Max_Right*

(e) Add and verify an iterative implementation for *Move_to_Max_Top*