**IIT CS536: Science of Programming**

Homework 3: Hoare triples and proofs

Prof. Stefan Muller

Submitted by: Abdurakhmon Urazboev

**Task 1.1**

1. this total correctness triple is not satisfied in the given state as it will not terminate. Total correctness has a condition that it should terminate, and the post condition should be true.
2. this partial correctness triple is satisfied in the given state as the precondition is met and if we iterate through the while loop after the first iteration, we get the state in this state program terminates as does not hold anymore. Now we know that precondition is met, and program terminates, meaning we should check if post condition is met after running the program. and the post condition is also correct for this state
3. partial correctness triple is satisfied because precondition is not met. In state , is false so the whole assumption is pointless, but the partial correctness is satisfied.
4. this partial correctness triple is satisfied in the given state as precondition is met the program runs to completion after the first iteration resulting in a state . If the program terminates, we should check if the post condition holds to see if the triple is valid. The post condition is . This is true as . So, this partial correctness triple is satisfied.
5. this partial correctness triple is satisfied in the given state. First, we check the precondition, and it is satisfied. Then we see if the program terminates, if not the triple is satisfied otherwise the post condition should be true for the triple to be satisfied. Program does not terminate as grows significantly faster than program will not reach a point of termination.

**Task 1.2**

1. this partial correctness triple is not valid because in a state precondition is true, and program terminates but the post condition is not satisfied as we see from the 1.1.b example above it terminates with state so post condition is not satisfied . To fix this triple, we could change the precondition to to avoid getting multiplied by negative numbers or 0. If we make both and negative numbers, the loop will stop with negative value being less than the value of , so we will stick with positive values only. With this precondition the triple will be valid as it will either terminate with value or does not terminate as will keep growing faster than . Either way, partial correctness triple will be valid.
2. this partial correctness triple is not valid because if we take random state { which satisfies the precondition and program runs and terminates successfully but the post condition will not be satisfied as program terminates with state and . To fix this we can limit the value of x such that program terminates, so we can change the precondition to this precondition makes sure that the program only runs for value and terminate. The postcondition will be satisfied as well as we saw in the 1.1.d. For values program will not satisfy the of while condition so values will not change and postcondition will hold.
3. this is not valid total correctness triple because it will not terminate. will keep growing linearly while will grow significantly faster. To fix that, we can change the statement s to run fixed times by changing to in the if statement. This way, it will run k-1 times and terminates with results that satisfy

**Task 1.3**

in order to come up with a solution we should start from the end and layer by layer analyze the triple. The postcondition and the statement indicate that r is positive number as power of any number is positive and initial value of r is either as at the end r is the multiple of 3. Number of multiplications should result in positive number. From that we can deduce that m should is a negative number. Let see if it is even or odd. If we take case we will have as a result as we can come up with initial value of r. so in order to satisfy this triple r should be 1 when m is negative even number. Let’s try again for odd numbers with . In this case r will evaluate to and . Now let’s find out a initial value of r for this case . Now we can come up with a condition that will make this triple valid

**Task 1.4**

To come up with precondition that makes valid, as this is assignment operation, we should make sure that right side of the equation is free of errors to make sure it runs and terminates successfully. So, as we are calculating root of it cannot be negative number so we add ghost variable to show that is non-negative number and as we are dividing it by y, y cannot be equal to 0 as we cannot divide by 0, . So, the precondition should be . This makes sure program does not end up in bottom state and postcondition T is always satisfied. So, the triple should look like as follows: .

**Task 1.5**

1. in order for a total correctness triple to be valid, if the precondition is satisfied then program terminates in certain state and postcondition is satisfied must hold. So, if the precondition is not satisfied then triple is considered valid as the statement does not say anything if precondition is not met. So, our precondition should be satisfied to make the triple invalid. As postcondition is defined as true, the only thing we can change here is that program should not terminate or should result in error. In that case the triple will be invalid. Conclusion: should be valid and should not terminate or result in bottom . The example would be . This triple is not valid as in any condition as root of a negative number cannot be calculated will result in an error.
2. Yes, we can create one with while clause, the approach will be the same. We should have precondition that is always satisfied and program with while clause that will not terminate. The example would be this triple is not valid as program will never terminate.

**Task 1.6**

1. A triple that is valid if and only if the program terminates when the initial value of x is greater than three. So, this triple should not be valid when it’s not terminated meening its total correctness triple. To do that we need to keep the initial value of x in ghost variable k and in the postcondition, we should specify that k is greater than 3. This way this triple will be valid if and only if it terminates when the initial value of x is greater than 3.
2. The triple which is valid if and only if program does not terminate when the initial value of x is less or equal to 3. This means triple should be valid when it’s not terminated so, it’s partial correctness triple. .

**Task 2.1**

**Task 3.1**

1. 1. Assign

2. Consequence (1)

3. Assign

4. Consequence (3)

5. If (2,3)

1. 0.

1.

2.

3.

4.

5.

**Task 3.2**

1. Assign
2. Assign
3. Assign
4. Simplify (1)
5. Simplify (2)
6. Simplify (3)
7. Consequence (5)
8. Consequence (6)
9. If(7,8)
10. If(4,9)