Hoare Logic Assignment

MUGOYA DIHFAHSIH
Student No. 2100702353 - Reg No. 2021/HD05/2353U

1. Are the following specifications partially correct?
   - Hoare logic is a way of characterizing program correctness
   - Checking for partial correctness
   - A program is said to be partially correct if it gives the right answer whenever it terminates.
   - {P} S {Q} => this is partially correctness specification because it does not require S to terminate
   - {P} is a precondition
   - {Q} is a post-condition
   - S is a program statement.

(a) {x = 1}y := x{y = 2}

Solution
Let us use Hoare Triple of form {P} S {Q}

```
        {x = 1}  y := x   {y = 2}
                {P}    S       {Q}
```

At the precondition x is equal to 1 at execution we substitute x with 1 and therefore y:=1 and the post-condition does not hold for y thus making the program **not correct**

(b) {x = a ∧ y = b}x := y; y := x{x = b ∧ y = a}
Solution
Precondition is {x = a ∧ y = b}
Statement s1 is x := y
Statement s2 is y := x
Post-condition is {x = b ∧ y = a}
The specification is partially correct because x and y in preconditions and after assignment results into a true post-condition.
The assignment of x:=y and y:=x maps values of a and b to the post-condition in a way that makes the specification hold when it terminates. Therefore it is a valid triple specification.

(c) {True}r := x; t := 0;WHILEy ≤ rDO(r := r − y; t := t + 1){r < y ∧ x = r + (y ∗ t)}

Solution
Precondition is {True}
Statement is r := x; t := 0;WHILEy ≤ rDO(r := r − y; t := t + 1)
Post-condition is){r < y ∧ x =r + (y ∗ t)}

- Since the precondition is true that implies the invariant of the loop is also true
- The invariant is preserved by the body of the while loop
- The invariant of the specification and the termination condition are true which implies that the post-condition is true.

Since the loop satisfies the three hoare triple checks, and with the fact that the post-condition is always true if the precondition is true whenever the loop terminates. This implies that the specification is partially correct.

2. Find the precondition assuming the statement x := x+b+1 executes and terminates in a state satisfying (b = 2) ∧ (x = y + b).
Using Hoare triple of the form {P} S {Q}

Precondition {P} is ??
Statement S is x:=x+b+1
The post-condition {Q} is {(b = 2) ∧ (x = y + b)}
Proving backwards
Replace x and b using the assignment
{(b=2) ∧ (x+b+1=y+b)}
{(b=2) ∧ (x=y-1)} this is the precondition that satisfies (b = 2) ∧ (x = y + b) when executed.

3. What is the suitable precondition for the code; x := x + 1; x := x ∗ x to establish a post-condition {x ≥ 16}

Using Hoare triple of the form {P} S {Q}
The precondition {P} is ?
Statements S x := x + 1; x := x ∗ x
Post-condition {Q} ={x ≥ 16}
Substituting the assigned value of x in post-condition
{(x + 1 ≥ 16) ∨ (x ∗ x) ≥ 16}
{P} results into {(x ≥ 15) ∨ (x ≥ 4)} which is the precondition

$\{(x \geq 15) \vee (x \geq 4)\}$ when executed using the assignments $x := x + 1; x := x * x$ satisfies $\{x \geq 16\}$

4. $\{P\}$ if $(i \leq j)$ then $m := i$ else $m := j$ $\{(m \leq i \wedge m \leq j) \wedge (m = i \vee m = j)\}$ What is the weakest precondition $\{P\}$

Solution

Structuring the specification as a program

```
$\{P\}$
If $(i \leq j)$
    $m := i$
else
    $m := j$
$\{(m \leq i \wedge m \leq j) \wedge (m = i \vee m = j)\}$
```

Let $Q=(m \leq i \wedge m \leq j) \wedge (m = i \vee m = j)$

Let wp be the weakest precondition

$wp(IF, Q) = (i \leq j \wedge wp(m := i, Q)) \vee (i \geq j \wedge wp(m := j, Q))$

Substitute m with i in Q and then in OR side substitute m with j in Q

$= (i \leq j \wedge (i \leq j \wedge i=j) \vee (i \geq j \wedge (j \leq i \wedge j=i)))$

$= (i<j \wedge i<j \vee i>j \wedge j<i)$

5. Prove partial correctness of the following program

```
$\{x >= 0 \wedge y >= 0\}$
    $a := 0;$
    $b := x;$
WHILE $(b \geq y)$ DO
    $b := b - y;$
    $a := a + 1$
od:
    $\{x = a * y + b \wedge b \geq 0 \wedge b < y\}$
```

Solution

To prove the correctness of a loop, we need to show that when the loop terminates, the post condition is satisfied regardless of how many times the loop ran. This can be done by proving the loop Invariant (I).

Loop variant is the precondition and post-condition of the loop body and it is always true at both start and end of the loop.

We need to show that

1. {I} holds immediately before the loop (at initialization)
2. That {I} holds at the end of the loop body given that $\{I \wedge (b \geq y)\}$ hold at the beginning of the loop body.
3. That $\{I \wedge !(b \geq y)\}$ implies post condition {Q}

We add these assertions to our generalized loop of

```
Precondition: $\{x >= 0 \wedge y >= 0\}$
      $a := 0;$
      $b := x;$
Loop Invariant {Inv}
WHILE $(b \geq y)$ DO
      $b := b - y;$
      $a := a + 1$
      Loop Invariant {Inv}
Post- Condition: $\{x = a * y + b \wedge b \geq 0 \wedge b < y\}$
```

$\{I \wedge !(b \geq y)\}$ → $\{x = a * y + b \wedge b \geq 0 \wedge b < y\}$

Our loop invariant (**Inv**) is $b \geq 0$

Now we substitute loop variant value in the specification above

```
$\{x >= 0 \wedge y >= 0\}$
      $a := 0;$
      $b := x;$
Inv: $( b \geq 0 )$
WHILE $(b \geq y)$ DO
      $b := b - y;$
      $a := a + 1$
      Inv: $( b \geq 0 )$
$\{x = a * y + b \wedge b \geq 0 \wedge b < y\}$
```

Before the while loop begins to execute, our loop invariant is True, the value of b must be either 0 or greater than zero for the loop to execute. After execution when the program terminates, b will be 0 or greater than zero, this makes our while loop hold of correct