Chun-Wei Chen
CSE 331
Homework 1
04/08/13

Problem 1

a.

```
{ x > 0 }
x = 10;
{ x = 10 }
y = 2 * x;
{ x = 10 & y = 20 }
z = y + 4;
{ x = 10 & y = 20 & z = 24 }
x = z / 2;
{ x = 12 & y = 20 & z = 24 }
y = 0;
{ x = 12 & y = 0 & z = 24 }
```

b.

```
{ x > 0 }
y = x;
{ x > 0 & y > 0 }
y = y + 2;
{ x > 0 & y > 2 }
```

c.

```
{ |x| > 10 }
x = -x;
{ |x| > 10 }
x = x / 2;
{ |x| > 5 }
x = x + 1;
{ x > 6 | x < -4 }
```

d.

```
{ y > 2x }
y = y * 2;
{ y > 4x }
x = x + 1;
{ y > 4x - 4 }
```

Problem 2

a.

```
{ x > 0 }
x = x + 5;
{ x > 5 }
```

```
   y = 2 * x;
   { y > 10 }
```

b.

```
   { x ≤ -3 }
   y = x + 6;
   { x + y ≤ 0 }
   z = x + y;
   { z ≤ 0 }
```

c.

```
   { 2x > w - 10 }
   y = w - 10;
   { 2x > y }
   x = 2 * x;
   { x > y }
```

d.

```
   { s < 2 ∧ w > 0 }
   t = 2 * s;
   { s < 2 ∧ 2s + w > t }
   r = w + 4;
   { r > 2s + w ∧ 2s + w > t }
   s = 2*s + w;
   { r > s ∧ s > t }
```

Problem 3

```
   { x > 0 | (x < 0 & x ≠ -1) }
   if (x >= 0)
      { x ≠ 0 }
      z = x;
      { z ≠ 0 }
   else
      { x ≠ -1 }
      z = x + 1;
      { z ≠ 0 }
   { z ≠ 0 ∨ z ≠ 0 } => { z ≠ 0 }
   { z ≠ 0 }
```

Problem 4

```
   P: { true }
   if (x == 0)
   {
      x = x + 1; // S1
   } else {
      x = x * 2; // S2
```

```
    }
    Q: { x is even }
```
Willy's definition: wp("If C, S1 else S2", Q) = wp(S1, Q) ∨ wp(S2, Q)

wp(IF, x is even) = wp(x = x + 1, x is even) ∨ wp(x = x * 2, x is even)
= (x is odd) ∨ true
= true

If x = 0, which satisfies precondition { true }, and goes through the if block, x is odd after the if block, which doesn't satisfy the postcondition { x is even }; therefore, Willy's definition is incorrect.

Problem 5
```
    P: { x < -1 }
    if (x == 0)
    {
      x = x + 1; // S1
    } else {
      x = x * 2; // S2
    }
    Q: { x < 0 }
```
Willy's definition: wp(S1, Q) ∧ wp(S2, Q)

wp(IF, x < 0) = wp(x = x + 1, x < 0) ∧ wp(x = x * 2, x < 0)
= (x < -1) ∧ (x < 0)
= x < -1

If x = -1, which doesn't satisfy precondition { x < -1 }, and goes through the if block, x < 0 after the if block, which satisfies the postcondition { x < 0 }; therefore, Willy's definition doesn't give the weakest precondition.

Problem 6

a. { x ≥ 10 }

c. { x > 0 ∨ y > 0 }

d. { |x + y| > w }

Problem 7

a. Valid. If x < 0 is true, y ≤ 0 is always true after y = 2 * x is executed; therefore, it's a valid Hoare triple.

b. Invalid. If x = y, which satisfies P { x ≥ y }, the value of z will be 0 after the assignment, which doesn't satisfy Q { z > 0 }. To make it valid, just modify Q to be { z ≥ 0 }.

c. Valid. If x is even before the if block is executed, y will be even after the if block is executed (since the code y = x is executed); if x is odd before the if block is executed, y will be even after the if block is executed (since the code y = x + 1 is executed). Therefore, postcondition is { y is even } ({ y is even & y is even})

d. Valid. If x < 0 is true, it will always execute x = -1 since x < 0 => x < 100; therefore, x < 0 is always true after the code is executed if x < 0 is true (since x = -1 and -1 < 0).

Problem 8

a.

```
{ x > 0 } // P
y = x - 1;
{ 2y > 0 } // I1
z = 2 * y;
{ z > 0 } // I2
z = z + 1;
{ z > 1 } // Q
```

Precondition using backward reasoning, called P', is { 2x > 2 } => { x > 1 }, while P given in the problem { x > 0 }, which is weaker than the weakest precondition from the backward reasoning. For instance, if x = 1, which satisfies P { x > 0 } given in the problem, z = 1 after the code is executed, which doesn't satisfy Q { x > 1 }; therefore, the precondition is sufficient to guarantee the postcondition.

b.

```
{ 2x ≥ w }
y = w - 2;
{ 2x - 2 ≥ y }
x = 2 * x;
{ x - 2 ≥ y }
z = x - 2;
{ z ≥ y }
```

The precondition is sufficient to guarantee the postcondition since the precondition given in the problem is the same as the result the weakest precondition from the backward reasoning.

c.

Forward Reasoning:

```
{ y > 0 }
if (x == y)
   { x = y & y > 0 }
   x = -1;
   { x = -1 & y > 0 }
else
   { x ≠ y & y > 0 }
   x = y - 1;
   { x = y - 1 & y > 0 }
{ (x = -1 | x = y - 1) & y > 0 } => { x < y }
```

Backward reasoning:

```
{ (x = y & y > -1) | x ≠ y }
if (x == y)
   { y > -1 }
   x = -1;
   { x < y }
else
```

```
      { true }
      x = y - 1;
      { x < y }
  { x < y }
```

Although { y > 0 } is not the weakest precondition of this code by using backward reasoning, it’s sufficient to guarantee the postcodition since the postcondition from forward reasoning { (x = -1 | x = y - 1) & y > 0 } implies { x < y }. Another way to verify the correctness is that since the precondition { y > 0 } satisfies the weakest precondition from backward reasoning { (x = y & y > -1) | x $\neq$ y }, the postcondition { x < y } must be true because it’s verified by backward reasoning.

Problem 9

```
{ true }
if (x % 2 == 0)
  { x is even }
  y = x;
  { y is even & (y = x | y = x + 1) }
else
  { x is odd }
  y = x + 1;
  { y is even & (y = x | y = x + 1) }
{ y is even & (y = x | y = x + 1) }
```