CSE 331 Winter 2013 Homework Assignment 1

Directions:

- Due Tuesday, Jan. 15 by 11 pm.
- Turn in your work online using the Catalyst dropbox.
- You should turn in a single PDF file named hw1_answers.pdf.
- Your file should be no larger than 3MB. Scanned copies of hand-written documents are fine as long as they are legible when printed.
- Feel free to rewrite the problems and solutions on a separate sheet – you do not have to turn in these specific pages with the blanks filled in.
- You may use any standard symbols for "and" and "or" (& and |, V and ^, etc.)
- If no precondition is required for a code sequence, write {true} to denote the trivial precondition.
- You may assume that integer overflow will never occur.

1. **Forward reasoning with assignment statements.** Write an assertion in each blank space indicating what is known about the program state, given the precondition and the previously executed statements. Be as specific as possible. The first assertion in part (a) is supplied as an example.

   a.
   ```
   { x > 0 }
   x = 10;
   { x == 10 }
   y = 2 * x;
   {________________________________}
   z = y + 4;
   {________________________________}
   x = z / 2;
   {________________________________}
   y = 0;
   {________________________________}
   ```

   b.
   ```
   { x > 0 }
   y = x;
   {________________________________}
   y = y + 2;
   {________________________________}
   ```

   c.
   ```
   { |x| > 10 }
   x = -x;
   {________________________________}
   x = x / 2;
   {________________________________}
   x = x + 1;
   {________________________________}
   ```

d. { y > 2x }
y = y * 2;
{________________________________}
x = x + 1;
{________________________________}

2. **Backward reasoning with assignment statements.** Find the weakest precondition for each sequence using backward reasoning, and write the appropriate assertion in each blank space.

a. {____________________________}
x = x + 5;
{____________________________}
y = 2 * x;
{ y > 10 }

b. {____________________________}
y = x + 6;
{____________________________}
z = x + y;
{ z <= 0 }

c. {____________________________}
y = w – 10;
{____________________________}
x = 2 * x;
{ x > y }

d. {____________________________}
t = 2 * s;
{____________________________}
r = w + 4;
{____________________________}
s = 2*s + w;
{ r > s & s > t }

3. **Backward reasoning with if/else statements.** Find the weakest precondition for the following conditional statement using backward reasoning, inserting the appropriate assertion in each blank. Be sure to verify that the intermediate postconditions for the two cases imply the total postcondition, i.e. show that (Q1 | Q2) => Q.

```
{______________________________}
if (x >= 0)
        {______________________________}
        z = x;
        {______________________________}
else
        {______________________________}
        z = x + 1;
        {______________________________}
{z != 0 }
```

4. **Weakest conditions.** Circle the weakest condition in each set.

   a. {x == 20 } { x > 10 } { x >= 10 }

   b. { t == 2 } { t != 0 } { t > 0 }

   c. { x > 0 & y > 0 } { x > 0 | y > 0 }

   d. { |x+y| > w } { x+y > w }

5. **Hoare triples.** State whether each Hoare triple is valid. If it is invalid, explain why and show how you would modify the precondition or postcondition to make it valid.

   a.
   ```
   { x < 0 }
   y = 2*x;
   { y <= 0 }
   ```

   b.
   ```
   { x >= y }
   z = x – y;
   { z > 0 }
   ```

c. {true}

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

d. { x < 0 }

```
if (x < 100)
        x = -1;
else
        x = 1;
{ x < 0 }
```

6. **Verifying correctness.** For each block of code, fill in the intermediate assertions, then use them to state whether the precondition is sufficient to guarantee the postcondition. If it the precondition is insufficient, explain why and indicate where the assertions don't match up.
   (Hint: for assignment statements, use backward reasoning to find the weakest precondition that guarantees the postcondition, then see if the given precondition is weaker than the weakest precondition. For conditional statements (if), you may find a combination of forward and backward reasoning most useful. Follow the rules given in class for what assertion to insert at each point.)

   a. { x > 0 }

```
y = x – 1;
{____________________________}
z = 2 * y;
{____________________________}
z = z + 1;
{ z > 1 }
```

b. { 2x >= w }
y = w – 2;
{______________________________}
x = 2*x;
{______________________________}
z = x – 2;
{ z >= y }

c. { y > 0 }
if (x == y)
{______________________________}
x= -1;
{______________________________}
else
{______________________________}
x = y - 1;
{______________________________}
{ x < y }

7. **Write and prove code.** Write a block of code that calculates the smallest even number greater than or equal to x and stores it in y. In other words, y will be assigned either x or x+1. Assume x and y have already been initialized, and annotate your code with assertions before and after each statement to prove that it is correct. At the end of the block, it should be true that y is even and that y == x or y == x +1.