CSE 331 Winter 2022 Homework 1

Directions:

- Due Tuesday, January 11 by **11 pm**. Try not to use a late day on this assignment, but if you do, be sure you are aware of the course policy on late assignments.
- Turn in your work online using Gradescope. You should turn in a single pdf file. You can have more than one answer per page, but if you can, please try to avoid page breaks in the middle of a question. Insert page breaks between questions as needed. Scanned copies of hand-written documents are fine if they are legible.
- 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 |, $\wedge$ and $\vee$, etc.)
- If no precondition is needed for a code sequence, write {true} to denote the trivial precondition.
- Assume that:
  - all numbers are integers
  - integer overflow will never occur
  - division is truncating integer division (like in Java).
- Final answers should be simplified, within reason, but do not weaken any assertions or assumptions and be careful about prematurely simplifying intermediate assertions.

1. **Forward reasoning with assignment statements.** Find the *strongest* postcondition for each sequence by writing 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.

   In the assertions you write for this problem:
   - Only refer to the current state. Do not refer to previous values (no subscripts or "x_pre", etc.)
   - Simplify but **do not weaken** your assertions.

   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.

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

4. **Weakest assertions.** Circle the weakest assertion in each set. If some of the assertions are incomparable (neither weaker or stronger than each other), circle both of them and show that they are incomparable by giving two example states, one which satisfies the first assertion but not the second, and another that satisfies the second assertion but not the first.

   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 >= 10)
        y = x % 7;
else
        y = x-1;
{ y < 9}
```

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 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.)

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 equals x or y equals x +1.