**CSE 331 Autumn 2021 HW1**

General Rules:

- For logical operators, you may use words (e.g., “or”) or any standard symbols (e.g., “V”).
- Assume that
  - all numbers are integers
  - integer overflow will never occur
  - integer division rounds toward zero (as in Java) (Example: 5 / 2 = 2)
- Simplify but ***do not weaken*** your assumptions.
- Due October 11, 5:00PM on Gradescope.

1. **Forward reasoning with assignment statements.** Find the *strongest* postcondition by writing an assertion in each blank space indicating what's known about the program state, given the precondition and the previously executed statements. The first assertion in part (a) is supplied as an example. Additionally, you should rewrite your assertions to only refer to the current state of variables (i.e., do not use subscripts).

   a. {{ }}
   a = 1;
   {{ a = 1 }}
   b = a + 2;
   {{ _____ }}
   c = b / 2;
   {{ _____ }}
   a = c - b;
   {{ _____ }}
   d = a * b;
   {{ _____ }}

   b. {{ |t| > 1 }}
   t = -t;
   {{ _____ }}
   r = 4 - t;
   {{ _____ }}
   k = r - 3;
   {{ _____ }}

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

   a. {{ _____ }}
   a = a + 10;
   {{ _____ }}
   b = a / 2;
   {{ _____ }}
   b = b - 5;
   {{ b > -1 }}

   b. {{ _____ }}
   t = 1 - 2 * w;
   {{ _____ }}
   r = w + 2 * r;
   {{ _____ }}
   s = w + 7;
   {{ r > s and s >= t }}

   c. {{ _____ }}
   q = q - 1;
   {{ _____ }}
   r = q * q;
   {{ _____ }}
   z = r - q;
   {{ z > q }}

3. **Forward reasoning with if/else statements.** Find the *strongest* postcondition for the following conditional statement using forward reasoning, inserting the appropriate assertion in each blank.

```
{{ x != 1 }}
if(x >= 1) {
        {{ _____ }}
        y = 1 - x;
        {{ _____ }}
} else {
        {{ _____ }}
        y = 1 + x;
        {{ _____ }}
}
{{ _____ }}
```

4. **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) {
        {{ _____ }}
        y = -x;
        {{ _____ }}
} else {
        {{ _____ }}
        y = 3 + x;
        {{ _____ }}
}
{{ y < 3 }}
```

5. **Weakest conditions.** For each of the following subproblems, compare all three assertions. If one assertion is *weakest*, circle it (or otherwise mark it). Otherwise, if some of the conditions are incomparable (neither weaker nor stronger than each other), circle both of them and demonstrate that they are incomparable by ***giving two example states*** that each satisfy one assertion but not the other.

| | | | |
|---|---|---|---|
| a. | {{ x < 1 }} | {{ x < 2 }} | {{ x != 2 }} |
| b. | {{ \|b\| > 6 }} | {{ b > 7 }} | {{ b < -6 }} |
| c. | {{ \|y\| > 1 }} | {{ if y <= 1, then y < -1 }} | {{ (y-1) * (y+1) >= 0 }} |
| d. | {{ a > 0 and b <= 0 }} | {{ if a > 0, then b <= 0 }} | {{ a > 0 or b <= 0 }} |

6. **Hoare triples.** State whether each Hoare triple is valid. If it is invalid, ***give an initial program state as a counterexample***.

   a.
   ```
   {{ h > 3 }}
   j = 1 - 2 * h;
   {{ j < -6 }}
   ```

   b.
   ```
   {{ a < -4 }}
   b = a / c;
   {{ b * c < -4 }}
   ```

   c.
   ```
   {{ }}
   if(x > 2) {
           y = x / 2;
   } else {
           z = 4 - x;
           y = z / 3;
   }
   {{ y > 0 }}
   ```

   d.
   ```
   { z < 0 }}
   if(x == -z) {
           x = x * z;
           z = z - 1;
   } else {
           x = z + 1;
           z = x / 2;
   }
   {{ x <= z }}
   ```

7. **Verifying correctness.** Fill in the intermediate assertions. Indicate with an arrow or by typing next to the line of code, whether you are reasoning forward or backward. Finally, ***state whether the code is correct***, i.e., whether the Hoare triples are *all* valid.

   a.
   ```
   {{ y <= 3x }}
   y = y - 3;
   {{ _____ }}
   x = 3 * x;
   {{ _____ }}
   x = x - 3;
   {{ y <= x }}
   ```

   b.
   ```
   {{ }}
   if(x <= 1 && -2 <= x) {
           {{ _____ }}
           x = x - 1;
           {{ _____ }}
           x = x * x;
           {{ _____ }}
   } else {
           {{ _____ }}
           x = x * x;
           {{ _____ }}
           x = 17 - x;
           {{ _____ }}
   }
   {{ x < 9 }}
   ```