CSE 331: Software Design & Implementation
Section 1 – Code Reasoning

For these problems, assume that all numbers are integers, that integer overflow will never occur, and that division is truncating integer division (like in Java).

1. Is the Hoare triple valid? If not, what is a counter example?
    a. {{ x is even }} x = x / 2 {{ x >= 1 }}

       Invalid, counterexample: x = 0

    b. {{ true }} y = x * 2 {{ y % 2 = 0 }}

       Valid

    c. {{ x > y and y != 0}} z = x / y {{ z >= 1 }}

       Invalid, x=1 and y = -1

    d. {{ x > 4 }} x = x / 2 {{ x > 2 }}

       Invalid, counterexample: x = 5

    e. {{ x = 5 and y > 20 }} z = y / x {{ z >= 4 }}

       Valid

2. Make the Hoare triple invalid by weakening the precondition or strengthening the postcondition
    a. {{ x is prime and x > 2 }} x = x + 1{{ x is even }}

       Example: {{ x is prime }} x = x + 1{{ x is even }}

    b. {{ true }} y = x * x {{ y >= 0 }}

       Example: {{ true }} y = x * x {{ y > 0 }}

    c. {{ z > 1 }} y = z * z {{ y != z }}

       Example: {{ z >= 1 }} y = z * z {{ y != z }}

3. For each pair of logical assertions, write a W next to the weaker assertion and S next to the stronger condition. If possible, point out an example included in the weaker assertion not included in the stronger one. If incomparable, indicate so.

    a. **{{ y > 23 }}** **{{ y >= 23 }}**

{{ y > 23 }} stronger, {{ y >= 23 }} weaker, ex: y = 23

b. **{{ y = 23 }}** **{{ y >= 23 }}**

{{ y = 23 }} stronger, {{ y >= 23 }} weaker, ex : y = 24

c. **{{ y < 0.23 }}** **{{ y < 0.00023 }}**

Equivalent because integers, both stronger and weaker

d. **{{ x = y * z }}** **{{ y = x / z }}**

Incomparable, look at (x=5, z=2,y=2) and (x=0,z=0,y=0)

e. **{{ is_prime(y) }}** **{{ is_odd(y) }}**

Incomparable, look at x = 2

f. **{{ is_even(y) }}** **{{ y % 2 = 0}}**

Equivalent, both are stronger and weaker

g. **{{ true }}** **{{ y > 0 }}**

{{ y > 0 }} is stronger, {{ y > 0 }} weaker, ex: y = -1

4. Fill in the blanks using **forward** reasoning.

```
{{ i >= 2 }}
x = 2 * i;

{{ x = 2 * i and i >= 2 }}
y = x;

{{ y = x and x = 2 * i and i >= 2 }}
z = (x + y) / 2;

{{ z = 2i and y = 2i and x = 2i and i >= 2 }}
```

5. Fill in the blanks using **forward** reasoning.

```
{{ x > 0 }}
x = 10;
```

Note: there was previously a typo here that said y = 10

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

{{ x = 10 and y = 20 }}
z = y + 4;

{{ x = 10 and y = 20 and z = 24 }}
x = z / 2;

{{ x = 12 and y = 20 and z = 24 }}
y = 0;

{{ x = 12 and y = 0 and z = 24 }}
```

6. Fill in the blanks using **forward** reasoning.

```
{{ x >= 0 ∧ y >= 0 }}
y = 16;

{{ x >= 0 and y = 16 }}

x = x + y;

{{ x >= 16 and y = 16 }}

x = sqrt(x);

{{ x >= 4 and y = 16 }}
y = y - x;

{{ x >= 0 and y = 16 - x }}
```