CSE 331: Software Design & Implementation
Section 2 – 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. Fill in the blanks using **forward** reasoning.

```
{ x >= 0 ∧ y >= 0 }
y = 16;

{ ____________________________________ }
x = x + y;

{ ____________________________________ }
x = sqrt(x);

{ ____________________________________ }
y = y - x;

{ ____________________________________ }

⇒ { ____________________________________ }
```

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

```
{ true }
if (x>0) {

     { ____________________________________ }
     y = 2*x;

     { ____________________________________ }
} else {

     { ____________________________________ }
     y = -2*x;

     { ____________________________________ }
}

{ ____________________________________ }

⇒ { ____________________________________ }
```

3. Fill in the blanks using **backward** reasoning.

```
{ _____ }
a = x + b;

{ _____ }
c = 2 * b - 4;

{ _____ }
x = a + c;

{ x > 0 }
```

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

```
{ _____ }
if (y > 5) {

     { _____ }
     x = y + 2

     { _____ }
} else {

     { _____ }
     x = y + z;

     { _____ }
}
{ x > 17 }
```

5. Fill in the blanks using **backward** reasoning (extra practice problems).

a.

```
{ _____ }
x = x - 2;

{ _____ }
z = x + 1;

{ z != 0 }
```

b.

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

{ _____ }
z = x + y;

{ z > 0 }
```

c.

```
{ _____ }
w = 2 * w;

{ _____ }
z = -w;

{ _____ }
y = v + 1;

{ _____ }
x = min(y, z);

{ x < 0 }
```

6. Prove that the following code computes the minimum.

```
{ true }
if (x < y) {

  { _____ }
  m = x;

  { _____ }
} else {

  { _____ }
  m = y;

  { _____ }
}
{ _____ }

{ m = min(x, y) }
```

7. For each pair of logical assertions, circle the **stronger** logical assertion (or indicate that the pair is **incomparable**).

a.	`{ y > 23 }`	`{ y >= 23 }`
b.	`{ y = 23 }`	`{ y >= 23 }`
c.	`{ y < 0.23 }`	`{ y < 0.00023 }`
d.	`{ x = y * z }`	`{ y = x / z }`
e.	`{ is_ prime(y) }`	`{ is_odd(y) }`

8. Verify that the following code correctly determines if x is power of 2.

{{ Precondition: x >= 1 }}

```
int k = 0;
int y = 1;

{{ Inv: y = 2^k and y/2 < x }}
while (y < x) {
  y = y * 2;
  k = k + 1;
}

{{ y = 2^k and y/2 < x <= y }}

if (y == x) {
  {{ Postcondition: x is a power of 2 }}
  return true;
} else {
  {{ Postcondition: x is not a power of 2 }}
  return false;
}
```