Problem 1 (6 pts, 1 pt each): Condition Strength

Indicate the **strongest** condition in each set. Write "None" if the conditions are unrelated by implication. Unless otherwise stated, assume all variables are ints.

```
1. { x is even && y == x + 1 }
{ x is even && y is odd }
2. { 0 ≤ x ≤ 5 }
{ 0 ≤ x ≤ 3 }
{1 ≤ x ≤ 3 }

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

4. {x is divisible by 50 }
   {x is divisible by 5 }

5. { -5 < x < 40 }
   { x = 3 }
   { 10 < x < 50 }

6. abs(result*result - x) ≤ 0.0001 assume result is a double
abs(result*result - x) ≤ 0.000001
```

Problem 2 (8 pts, 2 pts each): Hoare Triples

State whether each Hoare triple is valid. If it is invalid, explain why and show how you would modify the the postcondition to make it valid.
Unless otherwise stated, assume all variables are ints.

```
1.  { x > 0 }
    y = -2 * x;
    { y <= 0 }
valid

2.  { x + 1 <= N }
    x = x + 1
    { x <= N }
valid

3. { i + j != 0 }
    i = i + 1;
    j = j - 1;
    { i + j == 0 }
invalid. change postcondition to { i + j != 0 }

4. { true }
    if (x > y)
        m = x;
    else
        m = y;
    { m == x && x > y } || { m == y && y < x }
invalid. change postcondition to { m == x && x > y } || { m == y && y <= x }
```

Problem 3 (2 pts, 1 pt each): General Hoare Triples

```
   B, C, D, E, F are logical conditions (logical formulas).
   The following are true:
   B => C (B implies C, i.e., B is stronger than C)
   C => D
   E => F
   {C} code {F}

B           E
C   code    F
D

   Indicate whether the following are valid or possibly invalid.

  1.    {B} code {F}     valid
  2.    {B} code {E}     possibly invalid
```

Problem 4 (11 pts, 1 pt. for each condition): Forward reasoning

Find the **strongest postcondition** of each code sequence by inserting the appropriate condition in each blank. The first condition in part (a) is supplied as an example. Please simplify your answers as much as possible. Assume all variables are ints. Copy all code to your answer file and fill in the blanks. Carry all variables forward. Show all work.

```
1. { x > 0 }
      x = 10;
            { x == 10 }
      y = 20 - x;
            { x == 10 && y = 10 }
      z = y + 4;
            { x == 10 && y = 10 && z = 14 }
      y = 0;
            { x == 10 && y = 0 && z = 14 }

2. { |x| > 11 }
      x = -x;
            { |x| > 11 }
      x = x * x;
            { x > 121 }
      x = x + 1;
            { x > 122 }


3. { |x| < 5 }
      if (x > 0) {
                { 0 < x < 5 }
           y = x + 2;
                { 0 < x < 5 && 2 < y < 7 }
       } else {
                { -5 < x <= 0 }
           y = x - 1;
                { -5 < x <= 0 && -6 < y <= -1 }
       }
            { |x| < 5 && ( 2 < y < 7 || -6 < y <= -1 ) }
```

Problem 5 (14 pts, 1 pt each condition): Backward reasoning

Find the **weakest precondition** of each code sequence by inserting the appropriate condition in each blank. Simplify your answers as much as possible. Assume all variables are ints.
The first condition in part (1) is supplied as an example.
Copy all code to your answer file and fill in the blanks. Answers must be expressed in the format `wp(expression, condition) = precondition.`
Show all work.

```
1.  { wp(-5,y > -2 * x) = y > 10 }
     x = -5;
         { wp(z = 2 * x + y, z > 0) = (2 * x + y > 0) = (y > -2 * x) }
     z = 2 * x + y;
       { z > 0 }
2.  { wp( if(...){...}, x > 1 || x < -3 ) = (x > 1) || (x < -3)}
     if (x > 0) {
       { wp(x = x + 6, x > 7) = (x + 6 > 7) = (x > 1) }
      x = x + 6;
     else {
       { wp(x = 4 - x, x > 7) = (4 – x > 7) = (-x > 3) = (x < -3) }
      x = 4 - x;
     }
       { x > 7 }
3.    { wp( if(...){...}, x > -3 || x > -1 ) = (x > -1)   }
      if (x > 4) {
         { wp(x = x - 3, x > 0) = ( x > 3 ) }
            x = x - 3;
      } else {
         { wp(if(...){...}, x > -3 || x > -1 ) = (x >= -4) && (x > -1) = (x > -1)}
            if (x < -4) {
                 { wp(x = x + 3, x > 0) = (x > -3)}
                   x = x + 3;
           } else {
                 { wp(x = x + 1, x > 0) = (x > -1)}
                   x = x + 1;
           }
      }
            { x > 0 }

 4.   { wp(x = y + 2, x > 2 * y – 1) = (y + 2 > 2*y – 1) = (-y > - 3) = (y < 3) }
        x = y + 2;
         { wp(z = x + 1, z > 2 * y) =  (x + 1 > 2 * y) = (x > 2 * y – 1)}
        z = x + 1;
         {  z > 2 * y}

 5.
        { wp(if(...){...}, x != 0 || x != -1) = (x > 0) || (x < -1) }
      if (x ≥ 0)
        { wp(z = x, z != 0) = (x != 0)}
         z = x;
      else
        { wp(z = x + 1, z != 0) =  (x + 1 != 0) = (x != -1)}
         z = x + 1;
        { z != 0 }
```

Problem 6 (8 pts, 1 pt each condition, 1 pt sufficient/insufficient): Verifying Correctness

For each block of code, fill in the intermediate conditions, then use them to state whether the precondition is sufficient to guarantee the postcondition. If the precondition is insufficient, explain why.

Hint: Use backward reasoning to find the weakest precondition that guarantees the postcondition and see if the given precondition is too weak to guarantee the postcondition. In other words, is the given precondition weaker than the weakest precondition?

Copy all code to your answer file and fill in the blanks. Answers must be expressed in the format `wp(expression, condition) = precondition`. Show all work. Assume all variables are ints.

```
1.    { x < 2 }
      { wp(z = x - 1,z < 0) = (x – 1 < 0) = (x < 1) }
   z = x - 1;
      { wp(w = -z, w > 0) =  (-z > 0) = (z < 0) }
   w = -z;
      { wp(w = w + 1, w > 1) = (w + 1 > 1) = (w > 0) }
   w = w + 1;
   {w > 1}

Sufficient or Insufficient:
     Insufficient because given precondition x < 2 is weaker (less restrictive)
     than x < 1; one more value is allowed by x < 2 than x < 1


2.    { (x == y && y > 0) || (y != x) }
      { wp(if(...){...}, (y > -1) || (true) ) = (true) }
   if (x == y)
               { wp(x = -1, x < y) = (-1 < y) = (y > -1) }
       x = -1;
   else
                { wp(x = y - 1, x < y) = (y – 1 < y) = (true) }
       x = y - 1;
   { x < y }

Sufficient or Insufficient:
     Sufficient
```