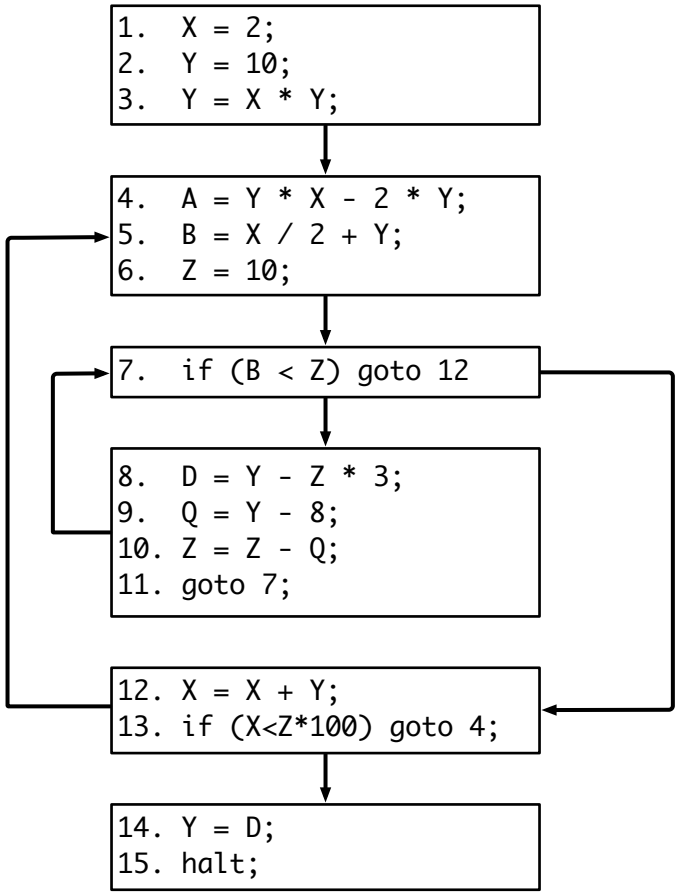
ECE 468 & 573
Problem Set 5: Loop transformations (Solutions)

Loop transformations
For the following problems, consider the code below:

```
1.   X = 2;
2.   Y = 10;
3.   Y = X * Y;
4.   A = Y * X - 2 * Y;
5.   B = X / 2 + Y;
6.   Z = 10;
7.   if (B < Z) goto 12
8.   D = Y - Z * 3;
9.   Q = Y - 8;
10.  Z = Z - Q;
11.  goto 7;
12.  X = X + Y;
13.  if (X < Z*100) goto 4;
14.  Y = D;
15.  halt;
```

1. Draw the CFG for the code above. Identify the loops in the code.

   Answer:



I will "name" basic blocks (BBs) by the instruction that starts them.

There are two loops in this program:

(a) BB 7 is a loop header: it dominates BB 8, and there is a back edge back to 7. BBs 7 and 8 are in the loop.

(b) BB 4 is a loop header: it dominates BB 12, and there is a back edge back to 4. BBs 4, 7, 8 and 12 are in the loop.

2. Which statements are loop invariant? Can they be moved outside their enclosing loop? Show the code that results after hoisting any loop invariant code outside the loop.

Answer: Let us proceed in stages. Statement 9 is loop invariant in BB 8: Y is only defined outside the loop. Moreover, Statement 9 can be hoisted outside the BB 7 loop, creating statement 7'.

```
1.   X = 2;
2.   Y = 10;
3.   Y = X * Y;
4.   A = Y * X - 2 * Y;
5.   B = X / 2 + Y;
6.   Z = 10;
6'.  Q = Y - 8;
7.   if (B < Z) goto 12
8.   D = Y - Z * 3;
10.  Z = Z - Q;
11.  goto 7;
12.  X = X + Y;
13.  if (X < Z*100) goto 4;
14.  Y = D;
15.  halt;
```

We now see that statement 6' is *still* loop invariant, and can be hoisted outside the loop. Statement 6 is loop invariant, but it *cannot* be hoisted out of the loop. Why? Because Z is defined multiple times inside the loop. The final code we are left with is:

```
1.   X = 2;
2.   Y = 10;
3.   Y = X * Y;
3'.  Q = Y - 8;
4.   A = Y * X - 2 * Y;
5.   B = X / 2 + Y;
6.   Z = 10;
7.   if (B < Z) goto 12
8.   D = Y - Z * 3;
10.  Z = Z - Q;
11.  goto 7;
12.  X = X + Y;
13.  if (X < Z*100) goto 4;
14.  Y = D;
15.  halt;
```

3. Identify the induction variables in this code. Show the code that results after performing any possible strength reduction.

Answer: Z is an induction variable for the inner loop (because Q is loop invariant), and D is a *mutual* induction variable based on Z (because Y is loop invariant). We have to be careful about the signs, though! If it helps, think about line 8 as being: `D = Y + Z * -3` and line 10 as being: `Z = Z + (-1 * Q)`. That means that our increment of D' should be: `D' = D' + (-1 * Q * -3)`, or `D' = D' + 3 * Q`. Rewriting the inner loop gives us:

```
1.   X = 2;
2.   Y = 10;
3.   Y = X * Y;
3'.  Q = Y - 8;
4.   A = Y * X - 2 * Y;
5.   B = X / 2 + Y;
6.   Z = 10;
6'.  D' = Y - Z * 3;
7.   if (B < Z) goto 12
8.   D = D'
10.  Z = Z - Q;
10'. D' = D' + 3 * Q;
11.  goto 7;
12.  X = X + Y;
13.  if (X < Z*100) goto 4;
14.  Y = D;
15.  halt;
```

We also see that X is an induction variable for the outer loop (because Y is loop invariant), so both A and B are mutual induction variables of X. Rewriting the code gives us:

```
1.   X = 2;
2.   Y = 10;
3.   Y = X * Y;
3'.  Q = Y - 8;
3''. A' = Y * X - 2 * Y;
3'''.B' = X / 2 + Y;
4.   A = A';
5.   B = B';
6.   Z = 10;
6'.  D' = Y - Z * 3;
7.   if (B < Z) goto 12
8.   D = D'
```

```
10.  Z = Z - Q;
10'. D' = D' + 3 * Q;
11.  goto 7;
12.  X = X + Y;
12'. A' = A' + Y * Y;
12''.B' = B' + Y / 2;
13.  if (X < Z*100) goto 4;
14.  Y = D;
15.  halt;
```

4. Show the code after performing any possible linear test replacement.

   Answer: We can replace the test in line 7 by an equivalent test. But we again have to be careful. To rewrite the test `B < Z` to use D' instead of Z, we have to multiply both sides by -3: `-3 * B > -3 * Z`, then add Y to both sides: `Y - 3 * B > Y - Z * 3`, which we can then turn into: `Y = 3 * B > D'`:

```
1.   X = 2;
2.   Y = 10;
3.   Y = X * Y;
3'.  Q = Y - 8;
3''. A' = Y * X - 2 * Y;
3'''.B' = X / 2 + Y;
4.   A = A';
5.   B = B';
6.   Z = 10;
6'.  D' = Y - Z * 3;
7.   if (Y - 3 * B > D') goto 12
8.   D = D'
10.  Z = Z - Q;
10'. D' = D' + 3 * Q;
11.  goto 7;
12.  X = X + Y;
12'. A' = A' + Y * Y;
12''.B' = B' + Y / 2;
13.  if (X < Z*100) goto 4;
14.  Y = D;
15.  halt;
```

   Note that we can't remove the definition of Z in line 10 – it is being used in line 13. We *can* perform linear test replacement for line 13 and remove the increment of X, though:

```
1.   X = 2;
2.   Y = 10;
3.   Y = X * Y;
3'.  Q = Y - 8;
3''. A' = Y * X - 2 * Y;
3'''.B' = X / 2 + Y;
4.   A = A';
5.   B = B';
6.   Z = 10;
6'.  D' = Y - Z * 3;
7.   if (Y - 3 * B > D') goto 12
8.   D = D'
10.  Z = Z - Q;
10'. D' = D' + 3 * Q;
11.  goto 7;
12.  //X = X + Y;
12'. A' = A' + Y * Y;
12''.B' = B' + Y / 2;
13.  if (A' < Y * Z * 100 - 2 * Y) goto 4;
14.  Y = D;
15.  halt;
```

5. Give an example of a piece of code with a doubly-nested loop where loop *interchange* is *not legal*.

   Answer: As we've seen in our dependence analysis slides, any doubly-nested loop that has a (+, -) direction vector will preclude loop interchange.