Program:

```
int x = 10, y = 0;
co
      ## Thread A
      {
            while (x != y) {  ## a1 (a1R1; a1R2)
                  x = x-1;  ## a2 (a2R; a2W)
            }
            y = y+1;  ## a3 (a3R; a3W)
      }
//
      ## Thread B
      {
            <await (x == y) ;>  ## b1 (b1R1 and b1R2 done atomically)
            x = 8;  ## b2
            y = 2;  ## b3
      }
oc
```

Possible interleavings/outcomes:
T'A could execute completely before T'B, leaving x==0, y==1. T'B would then fail to terminate, since x and y will never be equal.
Final outcome: no termination, x == 0, y == 1.

T'A could set x=0 via the loop of a1/a2, and break from the while loop.
T'B could then execute b1, b2, and b3. T'A could then execute a3, incrementing y.
Final outcome: termination, x == 8, y == 3.
T'A could set x=0 via the loop of a1/a2, and break from the while loop.
B'T could then only execute b1. T'A could terminate, executing a3, then T'B would execute b2 and b3, overwriting x and y. Alternatively, b1, a3R, b2, a3W, b3 could execute, with the same outcome.
Final outcome: termination, x == 8, y == 2.
T'A could set x=0 via the loop of a1/a2, and break from the while loop.
T'B could execute b1 and b2, then the read component of a3 executes, b3 executes, and then the write component of a3 executes. Alternatively, b1, a3R, b2, b3, a3W could execute; or b1, b2, a3R, b3, a3W could execute, both with the same outcomes.
Final outcome: termination, x == 8, y == 1.

T'A could set x=0 via the loop of a1/a2.

T’B could then execute b1, b2, and b3, setting x==8 and y==2. T’A could then check the condition of a1, find it false, and continue to loop, eventually breaking from the loop when x==2==y and then incrementing y.
Final outcome: termination, x == 2, y = 3.
T’A could set x=0 via the loop of a1/a2.
T’B could then execute b1 and b2, setting x=8. T’A could then check the condition of a1, find it false, and continue to loop, eventually setting x==0. Either a3, b3 could then execute, or b3, a3, or a3 read, b3, a3 write.
Final outcome: termination, x == 0, y == 2.
Final outcome: termination, x == 0, y == 3.
Final outcome: termination, x == 0, y == 1.

T’A could set x=0 via the loop of a1/a2.
T’B could then execute b1 and b2, setting x==8. T’A could then check the condition of a1, find it false, and continue to loop. When x==1, b3 could execute, setting y==2. T’A can now never terminate, as y will always be higher than x, and the while loop will keep decrementing x indefinitely.
Alternative conditions for this, all beginning with T’A setting x==0 via the loop of a1/a2, followed by T’B setting x==8 by executing b1 and b2:

- Loop a1/a2 until x==2 and y==0, after a1 check fails b3 executes, now x==2 and y==2, a2 decrements x, so now x==1 and y==2, and the loop will continue forever.
- Loop a1/a2 until x==1 and y==0, after a1 check fails b3 executes, now x==1 and y==2, and the loop will continue forever.
- Loop a1/a2 until x==0 and y==0, after a1R1 but before a2R2 b3 executes, now a1 sees x==0 and y==2, and the loop will continue forever.

Final outcome: no termination, x == -infinity, y == 2.