/* Constants */
#define N 3    /* number of workers or threads */
#define WIDTH 9   /* image width */
#define HEIGHT 6   /* image height */

/* Variables */
short VCW  = 0   /* vertical column width = WIDTH/number of workers or processes */
short H[N]= 0;    /* A counter for number completed for each horizontal section*/
short V[N] = 0;  /* A counter for number completed for each vertical section */
bool Vflag[N] = false;  /* Flag indicating whether a worker has started transforming a vertical section */

/* The following code shows the task performed by each thread/worker. First, worker performs * horizontal transformation on section (id-1), when complete it jumps to C1. At C1, worker checks * if there is any vertical section that is not being processed by another worker (i.e Vflag[n] == 0) * if found, it checks if horizontal transformation has already been done for that vertical section * if yes it sets the Vflag for that section to true and jumps to C2. In C2 vertical transformation * is done for the section found in C1. When complete, it jumps back to C1. In C1, if all vertical * sections have been transformed (i.e Vflag[n] == 0, where n = 0..N-1) then it jumps to END, where * worker exits */

proctype worker(byte id) {
  byte col;
  byte p;
  bool flag;
  short vid;

  do
    :: (H[id-1] < WIDTH) -> H[id-1]++
    :: else -> goto C1

  od;

C1:

atomic {
    col = 1;
do :: col <= N ->
        :: (Vflag[col -1] == false) ->
            p = 1;
            flag = true;
do :: p <= N ->
            if
                :: ( H[p-1] < (VCW *col) ) ->
                    flag = false
                :: else -> skip
            fi;
            p++
            :: p > N -> break
        od;
    if
        :: (flag == true) -> Vflag[col - 1] = true;
            vid = col - 1;
            goto C2
        :: else -> skip
    fi
    :: else -> skip
    fi;
col++;
    :: col > N ->
        break
    od;
};

col = 1;
flag = true;
do :: col <= N ->
    if
        ::(Vflag[col-1] == false) -> flag = false
        :: else -> skip
    fi;
    col++
    :: col > N ->
        break
    od;

if
    :: (flag == true) -> goto END
    :: else -> goto C1
fi;
C2:

    do

        :: (V[vid] < HEIGHT) -> V[vid]++
        :: else -> goto C1

    od;

END: skip

} /* The following proctype asserts that at all times if V[m-1] is greater than 0
then H[0] ..H[N-1]
* is greater than (VCW*m)
* That is to prove that before vertical transformation starts on a vertical
section ( i.e V[m-1] > 0) ),
* horizontal transformation has taken place on on that section and previous
sections
* ( i.e H[n-1] > vertical column width * m, where n = 1..N ). */

active proctype monitor() {
    byte m;
    byte n;

    m = 1;
    do
        :: m <= N -
        n = 1;
        do
            :: n <= N -
            if
                ::(V[m-1] > 0) ->
                assert( (H[n-1] >= (VCW*m)) );
            ::else -> skip
            fi;
            n++;
        :: n > N -> break
        od;
        m++;
    :: m > N -> break
    od
}
/* The initial process starts N number of threads /workers and sets vertical column width
* VCW = WIDTH/N
*/

init {
    byte proc;

    VCW = WIDTH/N;

    atomic {
        proc = 1;
        do
            :: proc <= N ->
                run worker(proc);
                proc++
            :: proc > N ->
                break
        od
    }
}