```
real grid[0:n+1,0:n+1];
int HEIGHT = n/PR;   # assume PR evenly divides n
real maxdiff[1:PR] = ([PR] 0.0);

procedure barrier(int id) {
  # efficient barrier algorithm from Section 3.4
}

process worker[w = 1 to PR] {
  int firstRow = (w-1)*HEIGHT + 1;
  int lastRow = firstRow + HEIGHT - 1;
  int jStart;
  real mydiff = 0.0;
  initialize my strip of grid, including boundaries;
  barrier(w);
  for [iters = 1 to MAXITERS] {
    # compute new values for red points in my strip
    for [i = firstRow to lastRow] {
      if (i%2 == 1) jStart = 1;      # odd row
      else jStart = 2;               # even row
      for [j = jStart to n by 2]
        grid[i,j] = (grid[i-1,j] + grid[i,j-1] +
                     grid[i+1,j] + grid[i,j+1]) * 0.25;
    }
    barrier(w);
    # compute new values for black points in my strip
    for [i = firstRow to lastRow] {
      if (i%2 == 1) jStart = 2;      # odd row
      else jStart = 1;               # even row
      for [j = jStart to n by 2]
        grid[i,j] = (grid[i-1,j] + grid[i,j-1] +
                     grid[i+1,j] + grid[i,j+1]) * 0.25;
    }
    barrier(w);
  }
  # compute maximum difference for my strip
  perform one more set of updates, keeping track of the maximum
      difference between old and new values of grid[i,j];
  maxdiff[w] = mydiff;
  barrier(w);
  # maximum difference is the max of the maxdiff[*]
}
```

Figure 11.6 Red/black Gauss-Seidel using shared variables.