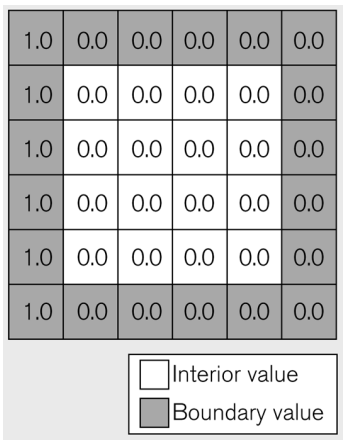
Comp. Arch. HW #6 Due: 11/1/14 (Sat.) at 11:59 pm

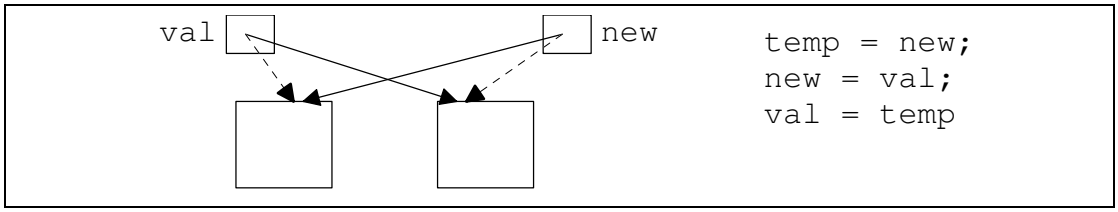
You are to complete the "starter" 2D Successive Over-Relaxation (SOR) (often used in 3D form to solve differential equations such as Navier-Stokes equations for fluid flow) program available from:

http://www.cs.uni.edu/~fienup/cs2420f14/homework/hw6.zip

Initially, the 2D-array `val` contains 0.0s everywhere, except for the 1.0s down column 0. On each iteration, SOR updates all **interior** values (i.e., only the white values in the diagram change with the gray boundary values being fixed) by the average of their four nearest neighbors. Eventually after many iterations the values will stablize. We won't run to complete stablization, but just until the maximum value change across the array during an iteration is less than a user specified *threshold* (e.g., 0.0001).



During an iteration use the 2D-array `val` to compute the updated values in the 2D-array `new`. Before the next iteration, flip-flop and reuse these 2D-arrays. This is actually easy and efficient since `val` and `new` are really pointers to the arrays, so all we really need to do is swap their pointer values.



You'll need to:

- decide how to decompose the work among threads -- keep in mind load balancing, cache performance
- decide how to synchronize the threads so all threads finish an iteration before any start the next iteration
- decide how to synchronize the threads so all threads stop if the max. change at any spot during an iteration is less than the threshold
- code the `thread_main` function run by all the threads
- test and debug your program

1) Download and extract the starter code `hw6.zip` which is available at:
http://www.cs.uni.edu/~fienup/cs2420f14/homework/

2) For this activity I want you to:

- use FileZilla, WinSCP, scp, ... to copy the starter code `hw6` directory to student.cs.uni.edu
- use an editor (emacs or nano) to complete the `hw6.c` program
- compile the C to an executable file using: `gcc -o hw6 hw6.c -lpthread -lm`
- when its working capture the interactive running of the program using: `script out.txt` to start the capture the timing by: `./hw6 1024 8 0.0001`
  to run and time the program on 1024x1024 interior array using 8 threads and threshold of 0.0001.
  Remember that `<Ctrl>+d` is used to end the script capture.

3) Use a secure ftp client (e.g., FileZilla, WinSCP, scp, etc.) to copy your `hw6` directory back to your local computer
(On a MAC you can probably use: `scp -r userName@student.cs.uni.edu:hw6 localDir`)

4) On your local computer zip the `hw6` directory and submit as Homework #6 at:
http://www.cs.uni.edu/~fienup/cs2420f14/homework/submissionDirections.htm

EXTRA CREDIT: Obtain "good" timings using i7 processor machines in ITT 335 by rebooting to Linux. You'll be the sole user of the CPU which has 4 cores that are hyperthreaded (runs two threads per core), so you see good speedup through 8 threads. Before timing, start the visual system monitor by Application | System Tools | System Monitor. Perform and report several timings for each number of threads using 1024 x 1024 internal array using 1, 2, 4, 8 threads.

Sequential code that performs the 2D SOR calculation:

```
void perform2D_SOR() {

  double average, maxDelta;
  double thisDelta;
  double **temp;
  int i, j;

  do {
    maxDelta = 0.0;

    // printVal();

    for (i = 1; i <= n; i++) {
      for (j = 1; j <= n; j++) {
        average = (val[i-1][j] + val[i][j+1] +
                   val[i+1][j] + val[i][j-1])/4;
        thisDelta = fabs(average - val[i][j]);
        if (maxDelta < thisDelta) {
          maxDelta = thisDelta;
        } // end if

        // store into new array
        new[i][j] = average;
      } // end for j
    } // end for i

    temp = new;     /* prepare for next iteration */
    new = val;
    val = temp;

    // printf("maxDelta = %8.6f\n", maxDelta);
  } while (maxDelta > threshold);  // end do-while

  delta = maxDelta; // sets global delta

} // end perform2D_SOR
```