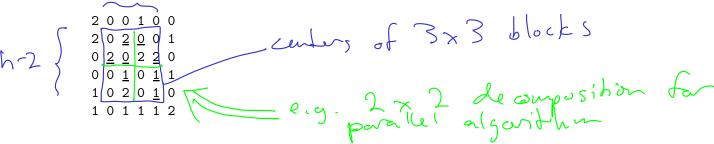
Question 1. [10 points] Assume that there is a grid of h rows and w columns of integer values. Also, assume that q is an arbitrary integer. Using pseudo-code, briefly sketch a *sequential* algorithm (i.e., not parallel) that will count how many 3x3 blocks of values sum to exactly q.

For example if q = 7 and the grid is



then the result of the algorithm should be 8. (The grid locations at the center of each 3x3 block whose sum is 7 are underlined in the grid shown above.)

Question 2. [30 points] Using pseudo-code, sketch an parallel algorithm for the problem described in Question 1. Your parallel algorithm should divide the overall h by w grid up into an N (rows) by M (columns) grid of processors, assigning a smaller local grid to each processor. Make sure your pseudo-code shows

• How each processor determines which portion of the overall grid it will work on

• How the local results computed by each processor are combined to form an overall solution proc-row = rank / M proc-col = rank 70 M pff_w= w-2 eff-h = h-2 chunk-size-horiz = eff_w/M startx = proc_col x chunk-size-horiz if (proc_col == M-1) { chunk_size_horiz += eff_w % M endx = startx + chunk-size-horiz chunk-size-vert = eff-h/N/ starty = proc-row of chunk-size-vert if (proc-row == N-1) { chunk_size-vert += eff_h76 N endy = starty + chunk_size_vert local_sum=0 for (i= starty+1; i < endy+1; i++) } for (j= startx + 1; j<endx+1; j++) {

Same as seq. alg. 1 local sum is the of 3 x 3 blocks in
1 local process summing to 9
All processes: reduce local sums by adding
to yield a single global sum

Question 3. [60 points] Implement your parallel algorithm using MPI. To get started, see the instructions on the exam web page:

```
http://ycpcs.github.io/cs365-spring2015/assign/exam01.html
```

Edit the code in countblocks.c. To run the program, use the command

```
./runpar filename \ q \ N \ M
```

where filename is an input file, q is the value of the integer q, N is the number of rows of processes, and M is the number of columns of processes.

The output of the program should be

```
Result is number
```

where number is the total number of 3x3 blocks whose sum was q.

Some hints and suggestions:

- Code to read the input data into a Grid object is provided
- Because the local processes don't modify any data (they just read the already-loaded data), you don't need to allocate a local Grid or copy any data into it: each process can use a region of the global Grid
- Use the grid_get_current function to get values from the global Grid
- You can use the divide_work function to help each process determine which ranges of rows and columns it should check (but you aren't required to use it)
- Only one process should report the overall result

Two test files are provided. Some tests you can try:

```
./runpar test1.dat 7 2 2
```

The output should be Result is 8.

Another test:

```
./runpar test1.dat 8 2 2
```

The output should be Result is 3.

Another test:

```
./runpar test2.dat 7 2 2
```

The output should be Result is 44.