In our first lab, we will write CUDA programs to determine the heat distribution in a space using synchronous iteration on a GPU. We have 2-dimensional square space and simple boundary conditions (edge points have fixed temperatures). The objective is to find the temperature distribution within. The temperature of the interior depends upon the temperatures around it. We find the temperature distribution by dividing the area into a fine mesh of points, \( h_{i,j} \). The temperature at an inside point is calculated as the average of the temperatures of the four neighboring points, as shown in the following figure.

The edge points are when \( i = 0, i = n, j = 0, \) or \( j = n \), and have fixed values corresponding to the fixed temperatures of the edges. The temperature at each interior point is calculated as:

\[
\tilde{h}_{i,j} = \frac{h_{i-1,j} + h_{i+1,j} + h_{i,j-1} + h_{i,j+1}}{4}
\]

\((0 < i < n; 0 < j < n, \text{ remember that edge points have fixed values})\)

Stopping condition: You can have a fixed number of iterations or when the difference between two consecutive iterations (calculated as average among all points) is less than or equal to some number \( e \). For this lab, we will use fixed number of iterations.

Assume we have \((n+1) \times (n+1)\) points (including edge points). The edge points are at 80F, except points \((0, 10)\) to \((0, 30)\) inclusive) have temperature of 150. Assume also that all internal points are initialized to zero.
What you have to do:

1. Write a sequential C version of the problem.
2. Write another CUDA version with all optimizations you can find.
3. Describe all the optimizations you have done in #2
4. Execute the programs in #1 and #3 with n = 100; 500; 1,000; 10,000; 100, 000; and 1,000,000. Assume 50 iterations.
5. Using `time` command, report the overall time taken (i.e. neither user nor system times). You may need to repeat the experiments 5 times and take the average to amortize any skew that may be caused by system load.
6. Draw a bar-graph showing n (x-axis) and the time (y-axis) for the 2 versions of the program (2 bars per n on the same graph).
7. What are your conclusions regarding:
   - When is GPU usage more beneficial (at which n and why)?
   - When is the speedup (i.e. time of CPU version / time of CUDA version) at its lowest? And why?
   - When is the speedup at its highest? And why?
8. Repeat #5, #6, and #7 with 500 iterations.
9. What is the effect of increasing the number of iterations?

What to submit:

- The source code you used for each program above. Name the source code files as `sequential.c`; and `cuda-version.cu`
- A report that contains the graphs and answers for questions mentioned.

Put all the above in a zip file named by your `lastname-firstname.zip` and email it to the grader.