**CSC 4760/5760, Spring 2023**

**Programming Assignment 3**

**Assign date: February 23, 2023 Due: March 11, 2023**

For this assignment, you will write an openMP heat transfer simulation.

**Problem Description**

We start with a room divided into a 2D grid. Inside the grid, we place a handful of “heaters” at various locations. A heater is a grid cell with a fixed temperature. The purpose of the simulation is to see what happens to the temperature in every grid cell as time progresses. The time is divided into discrete steps. At each time step, a new temperature is calculated for each grid cell based on the temperatures it its neighbors. Every cell has eight [neighbors](http://en.wikipedia.org/wiki/Moore_neighborhood), which are the cells that are horizontally, vertically, or diagonally adjacent.

We will use the following equation to determine the new temperature for a cell:

Tnew = (Told + (k\*(Ttop + Tbottom + Tleft + Tright + Ttop\_left + Tbottom\_left + Ttop\_right + Tbottom\_right))/8.0)/2.0

Tnew is the new temperature for a given cell.

Told is the previous temperature for a given cell.

Ttop, Tbottom, etc. are the previous temperatures for the given cell’s neighbors.

k represents the transfer rate. A large k will result in faster heat transfer. You can assume

**Implementation Guidelines**

You can use two 2D arrays of floats to store the cell values of previous time steps (input temperature grid) and current time steps (output temperature grid).

For each time step:

1. Set the “heater” cells back to their fixed temperature in the input grid. This is because we want them to remain constant.
2. Given the input temperature grid (the old values), compute the output temperatures for the output grid (the new values).
3. Copy the cell values from the output grid to input grid in preparation for the next time step.

At the beginning of the simulation you should set the cell values to the given base temperature.

\*For boundary cells (cells in the 1st and last row and 1st and last columns) some of the neighbors will be non-existent. For example, the 4 corner cells will have 3 neighbors and other boundary cells will have 5 neighbors instead of 8 neighbors. So in case of boundary cells use the base temperature value for the non-existent neighbors for computation.

**Interface**

Your program should have the following command line interface:

./heat\_transfer <num\_threads> <numRows> <numCols> <base temp> <k> <timesteps> <heaterFileName> <outputFileName>

heaterFileName is the name of a file specifying the locations and temperatures of the heaters.

k is the previously mentioned heat transfer rate (a float) use a value of k between 1 and 1.1

base\_temp is the initial temperature of the room.

time steps is the number of time steps.

outputFileName is the name of the file where the final room is stored.

The heater file will be in the following format:

numHeaters

row col temp

row col temp

…

row col temp

numHeaters is number of heaters, and thus the number of lines that follow.

Each subsequent line defines the location (row and col) and temperature (temp) of a heater.

**Example heater file**:

3

18 42 57.6

467 399 98.6

52 732 212.4

**Output:**

Grid written to a CSV.

**Report:**

Run the code for a grid size of 5000 by 5000 for 1000 time steps. Run your code using

4, 8, 16 and 32 threads. Run each thread count on 4, 8, and 16 cores, twice each .

(12 combinations, 24 total runs)

Write a report discussion the following points:

1. Partitioning strategy
2. Speedup, efficiency, and scalability with changing processor number
3. Comparison of execution times and speed up with the OpenMP implementation

**Submission and Grading**

Submit a zip file containing your code (source and header files). The zip should also contain the report in pdf format. Your report will count for 20% of your grade, and the program will make up the remaining 80%. Make sure you follow the given interface. Your programs should compile without errors or warnings. Your program should avoid runtime errors such as race conditions as well.

**Late Submissions**

Late submissions will be penalized 10 points per day for up to five days. Submissions more than 5 days late will receive no credit.

**Extra Credit (10 points)**

If a working terminal or GUI visualization is included, 10 additional points will be granted. Instructions on how to compile and run the visualization must be included.