Homework 2

Due by: July 31, 23:59

General Guidelines:

- This homework assignment includes also several .h and .cpp files. Read ahead.
- Your submitted code should be successfully compiled with "gcc -std=c++11 -o3 -o <binary filename>". No additional specific compilation flags are allowed.
- Make sure you use the exact file names, function names and class names mentioned below. Otherwise, you will face compilation errors during automatic testing.
- Don't use any external libraries for this homework, except TBB
- The homework assignment is 100% auto graded. You can resubmit your solution to gradescope multiple times before the due date of this homework assignment.
- In total, you should upload (together) two files to gradescope:
  - tree_barrier_impl.h (exercise #1)
  - heat.cpp (exercise #2)
- The automatic tests for this homework can run up to **few minutes**, per submission.

Exercise 1: a combining binary tree barrier – 30 points

The implementation of sense barriers discussed in class may suffer from high-memory contention. This is because it has the *counter* shared variable, updated by all threads entering the barrier. One way to reduce this contention (possibly at the cost of increased latency) is to use a combining paradigm like a combining tree discussed in class.

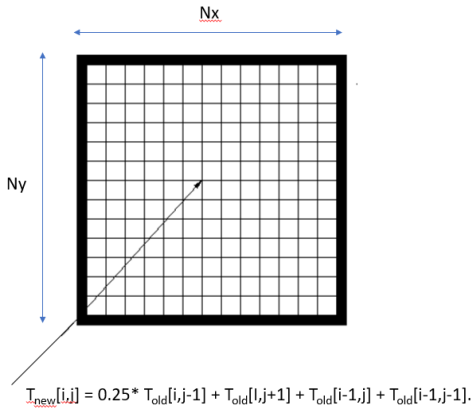
Implement a combining **binary** tree barrier, *BinaryTreeBarrier,* class which inherits the abstract class in *tree_barrier.h* file.

You can assume that the input parameter of number of threads to a binary tree barrier object is **always a power of 2.**

Upload *tree_barrier_impl.h* file with your binary tree barrier implementation to gradescope.

Exercise 2: heat 2D stencil computation – 40 points

A 2D heat diffusion models the propagation of heat from edges to middle of a plate by a stencil computation. Assume a plate is of size *Nx* by *Ny*. The temperature at the edge is a constant value *t_edge*. The temperature at each grid *point [i,j]* is initially set to zero and it is calculated iteratively as the average of the current temperatures at its four adjacent grid points. Iterations continue until the maximum change in temperature for any grid point is less than some threshold, *max_diff*, or when a limit of maximum number of iterations (defined by *max_iter*) is reached.



1. *heat_template.cpp* (attached with this homework assignment) currently provides a sequential implementation of this heat stencil. Get familiar with the code.
2. Notice this stencil computation differs from the stencil example discussed in class. In this case, updates of the internal heat elements (*point[i][j]*) are NOT done in-place.
3. Extend the code to run the heap stencil in parallel, using TBB patterns.
4. Under (mode==1), add a parallel implementation using *parallel_for* with a **global** *max_diff* variable to accumulate the convergence values.
5. Under (mode==2), add a parallel implementation using *parallel_for* with optimize mode==1 by using local *max_diff* variables to accumulate the local max convergence values for each thread before the global *max_diff* variable is updated.
6. Under (mode==3), add a parallel implementation using *parallel_reduce*.
7. Don't change the other parts of the code, as this may fail the automatic tests in gradescope.

Rename the *heat_template.cpp* file to *heat.cpp* and upload it to gradescope.

The automatic tests in Gradescope will check the correctness of the different mode implementations. Think hard how you can validate them before your submission. These automatic tests will measure the performance (execution time) of each mode and compare between them. Make sure you understand the expected performance behavior of the different mode implementations and validate your solution in advance before submission.

Good luck!