CS 380P Parallel Systems — Spring 2013	Programming Assignment #7
Distributed Work Queue in MPI	Due April 24, 2013

In this assignment, you will build a distributed work queue using MPI. To drive the design of your work queue, you will solve the Betweenness Centrality problem, which can be naturally implemented using a work queue. You may work with a partner if you wish.

1 The Betweenness Centrality Algorithm

The Brande's algorithm for finding Betweenness Centrality has parallelism at multiple levels. First, the shortest path exploration for each node can be done in parallel. Additionally, each of the shortest path computations can be done in parallel. You can get pseudo code for Brande's algorithm on page 10 of Brande's paper below. You can get more information about betweenness centrality on the Galois webpage:
`http://iss.ices.utexas.edu/?p=projects/galois/benchmarks/betweenness_centrality`

The following papers describe the algorithm in detail:

- Ulrik Brandes: A Faster Algorithm for Betweenness Centrality, *Journal of Mathematical Sociology*, 2001
- Edmonds, Hoefler, Lumsdaine: Space-Efficient Parallel Algorithm for Computing Betweenness Centrality in Distributed Memory, *IEEE*, 2010

2 Input/Output

Your program should accept the following parameters as input:

- Input Filename
- Output Filename

The Input file will follow the following pattern:

- Number of vertices (n), Number of Edges (e)
- e tuples specifying the edge between vertices.

To verify the result, your output should follow the following pattern:

- The vertices should be printed according to their index number.
- n tuples specifying the node index and its betweenness centrality

3 Test Inputs

Test Inputs will be provided by TA for this assignment. You should use those inputs to check the performance of your code on Stampede and Lonestar.

Assumptions. The test input will specify edges of an undirected unweighed graph, where edges are sorted in ascending order, taking (start vertex index, end vertex index) as a tuple.

What to Turn In

This assignment is due at 11:59pm on the due date. Use the **turnin** program to submit your solution, which should include the following:

1. A written report of the assignment in either plain or PDF format. This is your chance to explain your approach, your optimizations, any insights gained, problems encountered etc. Your report should include performance result for both Stampede and Lonestar.
2. Your source code, instructions to build it on Lonestar and Stampede, and instructions to run the programs. You may submit different codes for Lonestar and Stampede, in which case your report should explain why your codes differ for the two machines.
3. Further instructions will be provided by TA.