Eng 67/115: Lab 7

Goal: To explore the concepts of dynamic load balancing in *irregular* problems – problems where the data structures evolve at run time.

Required MPI Routines:

- `MPI_Init()`
- `MPI_Finalize()`
- `MPI_Comm_size()`
- `MPI_Comm_rank()`
- `MPI_Wtime()`
- `MPI_Recv()`
- `MPI_Send()`
- **MPI_Iprobe() % OPTIONAL**
- **MPI_get_count() % OPTIONAL**

TASK 1. Write a sequential program that prints *all solutions* to the N-queens problem – note that abstract code to solve this problem is presented in the appendix. A solution involves placing *N* queens on an *N* x *N* chess board such that no queen is in check (i.e. lies in the same row, column, or diagonal as another queen). For example, the *four*-queens problem has the following two solutions:

```
--Q-
Q---
---Q
-Q--
```

and

```
-Q--
---Q
Q---
--Q-
```

TASK 2. Write a concurrent program that prints all solutions to the N-queens problem where the value of N is taken as a command line argument. Have your program print the total number of queen placements in each computer as a measure of load-balance. Demonstrate that your program balances the load on each computer.

TASK 3 [ENG 115 ONLY]: Modify your manager worker code to:

1. Overlap communication and computation. This can be accomplished by modifying your worker to request the next piece of work just before beginning to work on the current piece of work. Note that you will have to terminate your code correctly.
2. Generate sub-problems in a separate thread. Ensure that you think through the case where generation of work is ***not*** complete but all work has been assigned to a worker!

APPENDIX:

In outline the sequential solution has the following form:

```
Queens()
{
            board[SIZE][SIZE]
            initialize_board
            place_queen(0,0)
}

place_queen(row, col)
{
    if(on_board(row, col)) {
        if(legal_move(row,col)) {
            board[row][col] = QUEEN
            place_queen(row+1, 0)
            board[row][col] = NOQUEEN
        }
        place_queen(row,col+1)
    }
}

legal_move(row,col)
{
    no queen in same col
    no queen in diagonal to left
    no queen in same diagonal to right
    if(no queen found)
        result = TRUE
    else
        result = FALSE
}
```