Problem Set 3
ISE 407 – Computational Methods in Optimization
Due: October 11, 2019
Dr. Ralphs

1. (a) Implement a simple sparse matrix class in C++ based on the compressed sparse row/column (CSR/CSC) format. Your class should allow for the user to choose whether the matrix is to be stored in CSR, CSC, or dense format. Your class should should implement the interface specified in a header file `SparseMatrix.h` in a file `SparseMatrix.cpp`. The specified implementation has a constructor as follows.

```
SparseMatrix(const int nCols,
             const int nRows,
             matrixFormat f,
             const double density,
             const bool initRandom);
```

   The `matrixFormat` type is an `enum` that indicates how the matrix should be stored. The `initRandom` parameter indicates whether the matrix should be randomly initialized and the density parameter is used in case of such random initialization or to indicate how much much space tpo allocate for nonzeros. You should have at least one method for adding elements/rows/columns to the matrix. It should be possible to do this dynamically (i.e., the size of the matrix should not be fixed).

   (b) Override the `*` operator so that two instances of your matrix class can be multiplied as in Matlab. It should be possible to multiply matrices of different formats. You should suypport `dense*dense`, `CSR*CSC`, `CSC*CSC`, and `CSR*CSR`.

   (c) Override the `<<` operator to provide a way of printing the matrix.

   (d) Use the provided driver to do some computational experiments comparing the empirical running time for multiplying random matrices of different densities stored both in sparse and dense formats. Study the empirical running time function for different densities, sizes, and data structures and report on your findings.

2. Define a *linear array of size n with a bus* to be a linear array augmented with a single global bus. Every processor is connected to the bus and in one unit of time, one processor can write to the bus and all other processors can read from it. This allows broadcasting in unit time, but only of one data word per time step.

   (a) State an efficient algorithm to sum n values, initially distributed one per processor, on such an architecture. What is the parallel cost of the algorithm? Compare to the case of finding the sum of n values on a regular linear array without a bus.

   (b) Can the parallel cost of the algorithm be improved by increasing the amount of data initially allocated to each processor?

3. Miller and Boxer, Chapter 5, Problem 1.

4. Miller and Boxer, Chapter 5, Problem 2.

5. Show that 3-COLOR is NP-complete by a reduction from 3-SAT and the decision problem in part (b) is therefore NP-complete. The initial construction goes as follows. Given a set of m clauses and n boolean variables $x_1, \ldots, x_n$ representing truth assignments, we construct a graph $G = (V, E)$ as follows. The set V consists of a vertex for each variable and a vertex for the negation of each variable, five vertices associated with each clause, and three special vertices: TRUE, FALSE, and RED. The edges of the graph are of two types: "literal" edges that are independent of the clauses and "clause" edges the depend on the clauses. The literal edges form a triangle on the special vertices and also a triangle on the vertices corresponding to x_i, $\bar{x}_i$, and RED for $i = 1, \ldots, n$.