Parallel Programming Assignment - 1*

Q1. **Rock/Scissors/Paper game**. Write a parallel program in Java to simulate a three-person rock/scissors/paper game. Each player randomly chooses one of rock, scissors, or paper. Then the players compare their choices to see who "won". Rock smashes scissors, scissors cut paper, and paper covers rock. Award a player 2 points if it beats both the others; award two players 1 point each if they both beat the third; otherwise award no points. Then the players play another game.

Use one thread for each player. The players must interact directly with each other. Do not use an additional coordinator thread. Print a trace of the results of each game as the program executes. At the end print the total points won by each player (the output should be just three numbers separated by a space, corresponding to the points won by each player). Read the number of games as a command line argument.

Q2. **Distribution of Matrix elements**: Given is a matrix of NxN integers, initialized to random values between 1 and 10. The problem is to calculate the *ultimate collapse* of values in rows, columns and all the elements. Write a parallel program in Java to solve the problem.

The collapse of a set of integers is defined to be the sum of the collapse values of the integers in the set. Similarly, the collapse of a single integer is defined to be the integer that is the sum of its digits. For example, the collapse of the integer 134957 is 29. This can clearly be earned out recursively, until a single digit results: the collapse of 29 is 11, and its collapse is the single digit 2. The ultimate collapse of a set of integers is just their collapse followed by the result being collapsed recursively until only a single digit 0, $1 \ldots 9$) remains. For row i, the output should be (ROW, i, c_i), if the collapsed value for the row i is c_i. For column j, the output should be (COL, j, c_j), if the collapsed value for the column j is c_j. The ultimate collapse value of the matrix should be printed as (MATRIX, c_m), if c_m is the ultimate collapse value of the matrix.

Read the size of the matrix and the number of threads as command line arguments. List the different patterns identified during "finding concurrency" and "algorithm design" phase.

*Assignments tweaked from that given by Greg Andrews@cs.Arizona and Manish Parashar@ece.rutgers

Evaluation criteria[1]:

1. 65% will be based on correctness - parallelizes correctly, creates correct number of threads, handles (in)correct command line arguments, gives the right answer.
2. 10% will be documentation - author's names, description of the program, how to use it, meaningful identifier names, meaningful function names, documentation for each function including arguments and return values.
3. 10% Source format - consistent indentation, readability, consistent capitalization, not too long lines (say 80 chars),
4. 15% Quality - Smartness of the solution (in terms of performance, used memory, algorithm and so on).

Java code to generate random numbers:

```
import java.util.Random;
...
  Random randomGenerator = new Random();
  int rInt = randomGenerator.nextInt(51);//generates a random number between 0-50.
```

[1]Criteria adapted from that of Peter Pacheco, University of San Francsico.