HOMEWORK ASSIGNMENT #6A

DUE: April 8, 2015
CSCI573: Probabilistic Reasoning, Prof. Nevatia
Spring Semester, 2015

This assignment will be handed out in two parts (6A and 6B); both due at the same time. This document describes 6A. It consists of one problem and requires programming.

Consider the "Alarm" Bayesian network from problem 2 of Assignment #3, with the given structure and CPTs. The task is also the same, computing probability of Earthquake given that JohnCalls is True and MaryCalls is False, but by using a sampling approach this time. Of course, the network is small enough that exact inference, by computing the joint distribution, variable elimination or belief propagation is easy; this assignment is thus just to practice with the tools of sampling which may be required when the networks get much larger and more complex.

Write a program to generate a Markov Chain by Gibbs sampling (Algorithm 12.4, page 506 in the KF book). Your program need not be general; it can be specific to the current network, even specific to the given evidence values. Note that the Gibbs sampler requires computation of the probability of a node given its Markov blanket; however, this is straightforward (see equation 12.23) and does not require you to implement a complex exact inference algorithm such as the sum-product algorithm.

To sample a variable with a given probability distribution, you will need to use a uniform random number generator; it is implemented in most programming languages. There are many (pseudo) random number functions in the standard C library. The simplest set of functions is:

```
int rand(void);
void srand(unsigned int seed);
```

`rand()` returns successive pseudo-random numbers in the range from 0 to (2^{15}-1)
`srand()` is used to set the seed. A simple example of using the time of the day to initiate a seed is via the call:

```
srand( (unsigned int) time( NULL ));
```

Run your program for enough number of iterations until a stationary distribution is achieved. You can test this by computing the frequencies of occurrence of the various combinations of the values for the "hidden" variables; note that you cannot compute these frequencies on a single iteration but need to consider an interval instead. It may be

instructive to notice how long it takes to achieve this balance. If you do not get convergence after a few thousand iterations, you may stop the algorithm anyway and report the results.

You should submit a copy of your code, a brief description of the key steps in it, the final probability distribution as well as output showing the functioning of your program. It is not practical or useful to give variable values after each iteration of the program; however, some illustrative examples and outputs after some number of iterations would be useful. It is left for you to decide the best way to describe the workings of your program but part of the grade will depend on your description, in addition to a correct program and answers.