Exercises: Embarrassingly Parallel Problems

1. **Simple Factorization**. The following pseudo-code shows a simple algorithm for finding all prime factors of a given number. Provide pseudo-code for a parallel version. Is your solution load-balanced?

   print_factors(n)
   // Print all prime factors of the input number
   1. for i $\leftarrow$ 2 to $\sqrt{n}$ by 1
   2. if i is prime
   3. while (n mod i = 0)
   4. print i
   5. n $\leftarrow$ n/i
   6. if (n = 1)
   7. break
   8. if n > 1 print n //last factor

2. **Growing Crystals**. Suppose we have a 3-dimensional lattice with n^3 points. To simulate crystal growth, we can compute a voronoi diagram. To do this we first choose m random chosen points as seeds. Then we walk through the lattice and at each point find the closest seed point. We label the point with the label of the seed point. After we have walked the whole lattice, then we write out the "crystal" to which each point belongs to a file. The following pseudo-code shows the implementation. Show how to change this pseudo-code to run in parallel. Also explain, how you would deal with the writing of a potentially very large output file.

```
numvel = n*n*n;
//volume is an array of numvel elements
choose_random_seeds(seed);  //seed[0..m-1]

for (vel=0; vel<numvel; vel++) {
    for (s=0; s<m; s++) {
        find distance of point number vel to the seed[s]
        update the closest found so far
    }
    volume[vel] = index of closest seed
}
```

```
write the volume array to a file
```

3. **Needle in a Haystack**. Suppose we have thousands of web server access logs that track user activity over the a few years. The format of the logs is shown below:

   ```
   5.55.209.124 - - [09/Dec/2007:04:48:24 -0700]
   "GET /~amit/teaching/453/quotes.html HTTP/1.0"
   200 10451 "-" "msnbot/1.0 (+http://search.msn.com/msnbot.htm)"

   66.249.65.198 - - [09/Dec/2007:04:52:53 -0700]
   "GET /~tim/courses/fall07/357/slides/Chapter%206/?C=M;O=D HTTP/1.1"
   200 988 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"

   74.6.19.76 - - [09/Dec/2007:04:53:10 -0700]
   "GET /%7Eamit/teaching/430/lab/monte_carlo.pdf HTTP/1.0"
   200 37315 "-" "Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysea
   ```

   Note that the first field is the IP address of the client requesting the page. The second field is the time stamp. The third field is the HTTP request. The next field is the HTTP code (200 is for successful request). The field after that is the number of bytes transferred. The last two fields are not relevant for our problem.

   Describe a parallel algorithm to find the user that has downloaded the most data from the web server. Assume that the log files are distributed evenly across the machines in the cluster.

4. **Load Imbalance Busters**. Suppose you are called upon to examine the performance of a nearly embarrassingly parallel program, which is suffering from poor speedup. You discover that the the program has a main loop that iterates n times. The number of processors is p, which is $\ll n$. Each iteration of the loop is independent. However, each iteration requires a variable amount of time that depends upon the input data. Suggest ways of load balancing this program.