CS121 Parallel Computing
Problem Set 4

1) Show that in a p-node hypercube, all the p data paths in a circular q-shift are congestion-free if E-cube routing is used.

2) Consider the following sequential *rank sort* algorithm for *n* values assuming no duplicate values:

```
for (i = 0; i < n; i++) {
    x = 0;
    for (j = 0; j < n; j++)
        if (a[i] > a[j]) x++;
    b[x] = a[i];
}
```

(a) Rewrite this as a parallel algorithm using OpenMP assuming that $p < n$ threads are used. Indicate clearly the use of private and shared variables and the schedule type.

(b) Modify the OpenMP code in part (a) to handle duplicates in the list of values, i.e. to sort into non-decreasing order. For example, the list of values [3, 5, 7, 5, 7, 9, 2, 3, 6, 7, 8, 1] should give the sorted list [1, 2, 3, 3, 5, 5, 6, 7, 7, 7, 8, 9].

3) The following sequential code calculates a triangular matrix using a function **`calc(i,j)`**, which has no data dependences and requires a constant (but large) amount of computation.

```
for (i = 0; i < n; i++)
   for (j = 0; j <= i; j++)
      a[i,j] = calc(i,j);
```

By inserting OpenMP directives into the sequential code, show how the following schemes for assigning work to threads may be implemented on a shared memory parallel architecture, commenting on the efficiency of each scheme:

a) A static block assignment of contiguous rows to threads.

b) A static cyclic assignment of single rows to threads.

c) A dynamic assignment of single rows to threads.

4) The *Back Substitution* algorithm solves a set of linear equations in upper (or lower) triangular form, as shown in Figure Q4a. A sequential algorithm to solve such a set of linear equations is given in Figure Q4b. Design a parallel algorithm for a shared

memory architecture and express it using OpenMP assuming that p < n threads are used. What schedule type would you use?

$$
\begin{aligned}
a_{n-1,0}x_0 + a_{n-1,1}x_1 + a_{n-1,2}x_2 \quad \ldots \quad + a_{n-1,n-1}x_{n-1} &= b_{n-1} \\
&\vdots \\
a_{2,0}x_0 + a_{2,1}x_1 + a_{2,2}x_2 &= b_2 \\
a_{1,0}x_0 + a_{1,1}x_1 &= b_1 \\
a_{0,0}x_0 &= b_0
\end{aligned}
$$

Figure Q4a

```
/* Back Substitution */
for (i = 0; i < n; i++) {
    x[i] = b[i]/a[i][i];
    for (j = i+1; j < n; j++) {
        b[j] = b[j] - a[j][i]*x[i];
        a[j,i] = 0;
    }
}
```

Figure Q4b