Name: _____

Directions: MAKE SURE TO COPY YOUR ANSWERS TO A SEPARATE SHEET FOR SENDING ME AN ELECTRONIC COPY LATER.

1. Consider the code in Sec. 1.5.2, which finds the maximum burst in a time series.

(a) (10) What will be printed out when **test()** is called?

(b) (10) What OpenMP construct is similar to the **if** statement prior to setting **rslts**?

2. The R **parallel** package includes a function **clusterApplyLB()**. It is similar to **clusterApply()**, but, in the docmentation's words,

> ...the first n jobs are placed in order on the n nodes. When the first job completes, the next job is placed on the node that has become free; this continues until all jobs are complete.

(a) (10) What does the abbreviation "LB" likely stand for?

(b) (10) What OpenMP keyword indicates an approach similar to this?

3. (30) The *(out)-degree* of a vertex i in a graph is the number of outlinks for that vertex. The OMP code below finds the degrees for the adjacency matrix **a**, returning them in **dgs** (preallocated by the caller). The number of vertices in the graph is **n**. Fill in the blanks.

```
void finddgs(int *a, int n, int *dgs) {
   blank (a)
   {  int i,j,dg;
      blank (b)
      for (i = 0; i < n; i++) {
         dg = 0;
         for (j = 0; i < n; j++)
            blank (c)
         dgs[i] = dg;
      }
   }
}
```

4. (30) In an interpreted language such as R or Python, the best speed often depends on making good use of built-in functions, such as matrix multiplication %*% in R. In this problem, we will explore an approach to speedy computation of the mutual outlinks application in Sec. 2.4.3. (A vertex might link to itself, but this will be irrelevant). We won't parallelize the operation here, but we could.

Fill in the blanks in the function **meanmo()** below. Note that blank (a) must be filled in with the name of a built-in R function, and that you should find the following R code helpful

```
> a <- rbind(1:2,3:4)
> a
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> row(a)
     [,1] [,2]
[1,]    1    1
[2,]    2    2
> row(a) == 1
      [,1]  [,2]
[1,]  TRUE  TRUE
[2,] FALSE FALSE
> sum(row(a) == 1)
[1] 2
> sum(row(a) > col(a))
[1] 1
> a[row(a) == col(a)] <- 8
> a
     [,1] [,2]
[1,]    8    2
[2,]    3    8
```

Here is the function:

```
meanmo <- function(a) {
   aap <- a %*% blank (a)(a)
   nummos <- blank (b)
   n <- nrow(a)
   nummos / blank (c)
}
```

Solutions:

1.a 9, 16

1.b single

2.a load balancing

2.b dynamic

3.

```
void finddgs(int *a, int n, int *dgs) {
   #pragma omp parallel
   {  int i,j,dg;
      #pragma omp for
      for (i = 0; i < n; i++) {
         dg = 0;
         for (j = 0; j < n; j++)
            dg += a[i*n+j];
         dgs[i] = dg;
      }
   }
}
```

4.

```
meanmo <- function(a) {
   aap <- a %*% t(a)
   nummos <- sum(aap[row(aap) < col(aap)])
   n <- nrow(a)
   nummos / (n*(n-1)/2)
}
```