This exam has ?? questions, for a total of ?? points.

Undergraduates may attempt 150 points out of 165 points. So they need to skip one of the problems worth 15 points. Graduate students need to attempt all problems.

Run \LaTeX{} again to produce the table.
1. **(20 points) Amdahl’s and Gustafson’s Laws.** Suppose we have a parallel program that is 5% serial and 95% linearly parallelizable for the given size. Assume that the serial part doesn’t grow as the problem is scaled. Then answer the following questions:

   (a) How much speedup do we get if we use 100 processors without scaling the problem? How much is the speedup if we have infinite processors.

   (b) Suppose we scale up the problem by 50. How much speedup would we now get with 100 processors?
2. (20 points) **Bag of Design Tricks.** Describe your bag of parallel algorithm design tricks.
3. (20 points) Three-dimensional Wrap-around Mesh Network. Calculate the diameter, cost and bisection width of a \( n = m \times m \times m \) 3-d mesh with wrap-around connections. Please write your answers in terms of \( n \).
4. **(15 points) Parallel, anyone?** Show how to convert the following sequential code to parallel?

```java
while (emptyQ() == false) {
    element = removeQ(); // queue functions take a small amount of time
    // processing each element takes a variable amount of time
    newelements = process(element);
    insertQ(newelements); // new elements is an array of objects
}
```
5. (15 points) **Better Broadcast.** Consider the following pseudo-code to perform a broadcast of an $n$ element array on a cluster with $p$ processes.

```c
//data[0..n-1]
if (pid == 0)
    for (i=0; i<n; i++)
        send (data, i); // send data array to the ith process
else
    recv (data, 0); //receive data array from process 0
```

The above code takes $p(t_{startup} + nt_{data})$ amount of communication time. Suppose we know that the cluster is connected via a switch. Then write a better broadcast code that takes time $\lg p(t_{startup} + nt_{data})$. (**Hint:** Think trees)
6. (15 points) MPI Code Reading at your local CodeStore. What does the following MPI code do?

```c
//assume that p = 2^k
//pid is the process rank [0..p-1]
for (i=0; i< k; i++) {
    if (pid < (1<<i))
        MPI_Send(buffer, n, MPI_INT, pid + (1<<i), TAG, communicator);
    if (pid >= (1<<i)) && (pid < 1<<(i+1))
        MPI_Recv(buffer, n, MPI_INT, pid - (1<<i), TAG, communicator);
}
```
7. (30 points) **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.

```plaintext
text
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
```
8. (30 points) **Game of Life.** A game where the board consists of a (theoretically-infinite) two-dimensional array of cells. Each cell can hold one "organism" and has eight neighboring cells, including those diagonally adjacent. Initially, some of the cells are occupied in a pattern. The following rules apply:

- Every organism with two or three neighboring organisms survives for the next generation.
- Every organism with four or more neighbors dies from overpopulation.
- Every organism with one neighbor or none dies from isolation.
- Each empty cell adjacent to exactly three occupied neighbors will give birth to an organism.

For the simulation, you may assume that the two-dimensional array is a torus so each cell always has eight neighbors. Sketch out how you would parallelize the $n \times n$ Game of Life on a $p$ processor cluster, where $p << n$. Does your implementation need to use a barrier. If so, where? Are there load balancing issues? Analyze the computation time and the speedup that your approach would yield.