1. What happens if in the pseudo-code we discussed in class for tree-structured global sum is run when the number of cores is not a power of two? Can you modify the pseudo-code so that it will work correctly regardless of the number of cores?

If we have our number of cores as a power of 2, then each core is assigned a unique bit of data and each step reduces our problem space by half. Each step requires half as many cores, thus, if we want to have an integer number of cores at each step, we must start with our number of cores as a power of 2.

If the number of cores is not a power of 2, we could pretend like we have the next power of 2 cores running, if a core which doesn't actually exist is supposed to send data to be added by another core which does exist, the actual core will simply assume a 0 was sent. Basically, we are extending our global sum to include an additional set of theoretically 0's such that we end up with a number of cores which is a power of 2. The pseudo code would look something like this.

n = number of actual cores
pow(2, ceil(log2(n))) // number of 'virtual' cores
// if core id $< n - 1$, take the partial sum as normal
// else (if id $= n - 1$), pretend to receive a 0 until it is our turn to send
// our data to the next core to do addition

2. Derive formulas for the number of receives and additions that core 0 carries out using:

(a) the original pseudo-code for a global sum

(b) the tree-structured global sum

(a) A total of n operations are needed, where n is the number of items to be added.

(b) The depth of the tree will be $\log_2(n)$ but this includes the initial value owned by core 0. Thus it receives and adds an additional $\log_2(n) - 1$ values to it's original value.

3. The first part of global sum example - when each core adds its assigned computed values - is usually considered to be an example of data-parallelism, while the second part of the first global sum - when the cores send their partial sums to the master core, which adds

them - could be considered to be an example of task-parallelism. What about the second part of global sum - when the cores use a tree-structure to add their partial sums? Is this an example of data- or task-parallelism? Why?

This is an example of data-parallelism, each core will be performing the same task (i.e. the same block of code), the only distinguishing factor is the tree node (the data) to which each core is assigned.