Homework #1 CS 1645/2045
Due: January 17, 2017 Spring 2017

Question 1: Assume that a task, T, is composed of two subtasks, T_s and T_p. Subtask T_s has an execution time of 2 Minutes, regardless of the number of processors available or size of the data input. Subtask T_p can be infinitely divided into any number of equal subtasks that can be executed in parallel. As such, subtask T_p execution times varies depending on the number of processors available and size of the data input, as follows:

Data Input Size	Processors	Execution Time
1GB	1	60 Minutes
1GB	2	30 Minutes
1GB	n	60 / n Minutes
10GB	1	240 Minutes
10GB	2	120 Minutes
10GB	n	240 / n Minutes

From this information calculate the following:

A. Speedup and efficiency for 1GB data input and 4 processors?
B. Speedup and efficiency for 10GB data input and 4 processors?
C. The maximum speedup for 1GB and 10GB?
D. For 10Gb data input what is the efficiency gain between having 1000 processors and 2000 processors?

Question 2: (Exercise 1.1 from the textbook) Devise a formula for the function that calculates my_first and my_last in the sum example of slide 19. Remember that, to achieve load balancing, cores should be assigned roughly the same number of elements of computation in the loop. Do not assume that n is evenly divisible by p. You can write a mathematical formula or code that produces these values.

Question 3: Consider the parallel sum algorithm described on slide 35 of the class presentation. Derive expressions for the speedup when the algorithm is applied to sum n numbers on p processors in the following two cases:

A. p is a power of 2 and n is a multiple of p.
B. The only restriction is that n > p.

Question 4: Assume that we have n processors working on the problem of finding the maximum value of a large set of numbers (much larger than n). After each n processors have found their maximum value (my_max) we need to find the overall maximum value. The following pseudocode takes n-1 time steps to compute the maximum of n numbers stored in n processors:

```
If (Pid == 0) { /* this part is executed by processor whose Pid = 0*/
    max = my_max;
    for (i=1 ; i < n ; i++) {
        receive maximum_value from processor i;
        if maximum_value > max {
          max = maximum_value;
        }
    }
} else { /* this part is executed by processors with Pid = 1, 2, … , n-1 */
    send my_max to the processor whose Pid = 0 ;
}
```

Using the same style, write the pseudocode for the tree-based structure for finding the maximum value of all n processors. This is very similar to the structure described in the slides for finding the sum. Assume that n is a power of 2.

Question 5: Repeat Question 4 assuming that the value of n is not necessarily a power of 2.