Instructions

- The solutions to all the problems in this homework should be the result of the student’s individual effort. Presenting the words or ideas of somebody else under your name hinders your skills in academic research and violates the university’s policy on academic integrity (http://www.provost.pitt.edu/info/ai1.html).

- Turn in a hardcopy of your solutions at the end of the session on Tuesday January 28th, 2014.

- Late submissions will not be accepted.

Problem 1

(a) If one worker can dig a post hole in one hour, can a dozen workers dig the post hole in five minutes? Why?

(b) If one worker can drive a railroad spike with a sledgehammer in two minutes, can two workers with sledgehammers drive the spike in one minute? Why?

(c) What accounts for the difference between the answers to parts (a) and (b)?

Problem 2

Consider the following code section of a parallel program that computes the sum of $n$ numbers in parallel using $p$ processors:

```c
my_sum = 0;
my_first_i = ... ;
my_last_i = ... ;
for(my_i = my_first_i; my_i < my_last_i; my_i++) {
    my_x = Compute_next_value(...);
    my_sum += my_x;
}
```

The code above computes the local sum of the numbers assigned to a particular processor. Devise a formula for the function that calculates `my_first_i` and `my_last_i`, assuming that each processor gets a section of a shared array storing the $n$ numbers. Remember that, to achieve load balancing, processors should be assigned approximately the same number of elements of the array. Do not assume that $n$ is evenly divisible by $p$. 

Problem 3

Assume that a task $T$, is composed of two subtasks, $T_0$ and $T_1$. The execution time of subtask $T_1$ on one processor is $t_1$ but it can be further divided into any number of equal subtasks that can execute in parallel. The execution time of subtask $T_0$ on one processor is $t_0$ and it cannot be further subdivided. Moreover, $T_0$ has to finish execution before $T_1$ can start execution.

(a) What is the maximum speedup and efficiency of executing $T$ on a system with $p$ processors? Assume the subtasks of $T_1$ can all run in parallel.

(b) Derive a formula for $S_{\text{max}}$, the maximum speedup that you can obtain even if you have available a very large number of processors.

(c) Plot $S_{\text{max}}$ as a function of $\alpha$, where $\alpha = t_0 / (t_0 + t_1)$ is the serial proportion of the computation that cannot execute in parallel. The parameter $\alpha$ ranges from 0 to 1.

Problem 4

Consider the parallel sum algorithm described on the Introduction to HPC lecture. 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 $n > p$.

Problem 5

A system with $n$ processors implements message passing as the only communication mechanism. There are no shared memory data structures. The following program illustrates how to add $n$ values (distributed one on each processor; stored in `my_x`). The program takes $n - 1$ time steps to compute the sum of $n$ values:

```c
if(ID == 0) { /* executed by processor with ID = 0*/
    sum = my_x;
    for (i=1; i < n; i++) {
        receive value from processor i;
        sum = sum + value;
    }
} else { /* executed by processors with ID = 1, 2, ... , n-1 */
    send my_x to the processor with ID = 0;
}
```

Using the same conventions, write the pseudo-code for a parallel sum that uses a spanning tree and takes $\log(n)$ steps. Assume that $n$ is a power of 2.