Due: Friday, 02/17/2023 at 11:59 pm

Instructions: You may work on this assignment in groups of up to 3 and submit a single solution for your group. All group members are responsible for understanding all submitted solutions.

Exercise 1.. Suppose a task T is composed of four (purely sequential) sub-tasks T_1, T_2, T_3, T_4 which require 360, 210, 210, and 180 elementary operations (respectively) to complete. We have three processors at our disposal P_1, P_2, P_3, which can perform 480, 320, and 320 operations per second (respectively).

(a) How could the tasks be assigned to processors so as to minimize the latency (i.e., completion time) of the overall computation? What is the latency of the computation?

(b) Suppose the speed of the three processors are still 480, 320, 320, but it is not known *which* of the processors is faster. A *greedy* schedule is a schedule for which whenever a process completes a task, the next task that has neither been completed nor is in progress is assigned to that process. What are the maximum and minimum latencies achieved by greedy scheduling for the three processes?

(c) Again in the case where it is not known which of the processors is fastest, show that there is a greedy schedule in which a non-greedy schedule would have lower latency. (A schedule is *non-greedy* if at some time, there is an idle process as well as a task that is neither completed nor in progess. That is, the task *could* be assigned to the process, but is not.)

Exercise 2. Suppose you are given a program with a method `M` that executes sequentially. Use Amdahl's law to answer the following questions, assuming that the remaining operations in the program can be computed in parallel.

(a) If `M` accounts for 20% of the program's total execution time, what is the maximum possible overall speedup on a 2 processor machine? A 3 processor machine? An 8 processor machine?

(b) What is the maximum possible speedup for *any* number of parallel processors (i.e., as the number n of processors becomes arbitrarily large)?

(c) Now suppose `M` is replaced by a method `N` that accomplishes the same task but in half the time. How much faster is the new program than the old program on a 2 processor machine? A 3 processor machine? An 8 processor machine? An n processor machine (as a function of n)?

Exercise 3. Consider the `Counter` object described in lecture:

```
1  public class Counter {
2    private long count = 0;
3
4    ...
5
6    public void increment() {
7      count++;
8    }
9  }
```

Suppose two threads concurrently increment a shared `Counter` object n times each. That is, both threads execute the following:

```
1  Counter ctr;
2  ...
3  for (int i = 0; i < n; i++) {
4    ctr.increment();
5  }
```

(a) For any given value c satisfying $n \leq c \leq 2n$, describe an execution for which the final value of `ctr` is c.

(b) Describe an exection for which the final value of `ctr` is 2.

(c) Is it possible that the final value of the counter could be 1? Why or why not?