The questions on page 1 are for the paper-and-pencil version of the exam. This version can be solved without computer. Text books, copies of slides, and notes are allowed. The decision to do the paper-and-pencil version of this exam can be postponed till 8:59PM, if you do not hand in at 9PM the answers to the questions below. In that case, it will be assumed that you do the use-computer version of the exam, with questions on page 2. Both versions of the exam must be solved individually. All questions are worth the same amount of points. Upload your answers to gradescope.

1. Suppose we have for every angle t at our current position, a function $d(t)$ which returns the distance to the closest object in that direction of the angle t. For a given $\Delta t > 0$, evaluate the function $d(t)$ at all angles from 0 to 2π, separated from each other by Δt. A convenient choice for Δt is $2\pi/n$, for some large positive integer n.

   (a) Design a parallel algorithm using the manager/worker paradigm for a distributed memory parallel computer to find the angle t for which the distance is smallest.

   (b) What is the best granularity for an optimal computation-communication ratio?

2. Given are two vectors $\mathbf{x}$ and $\mathbf{y}$, both of length n, where $y_k = f(x_k)$ for $k = 1, 2, \ldots, n$, for some function f evaluated at distinct points $x_i \neq x_j$ for $i \neq j$.

   The vector of divided differences is computed by the code below:

```
for i from 2 to n do
    for j from 1 to i-1 do
        y[i] = (y[j] - y[i])/(x[j] - x[i])
```

   (a) Define the task graph for a parallel computation of the vector of divided differences.

   (b) Do a critical path analysis on the graph to determine the upper limit of the speedup.

3. Given are n weights w_1, w_2, ... , w_n (given as positive doubles) and two doubles L and M, with $L < M$.

   The knapsack problem asks for all sums S of subsets of the weights with $L \leq S \leq M$.

   Each solution is represented as a set of indices, a subset of $\{1, 2, \ldots, n\}$.

   (a) Use the recursive formulation of the algorithm to generate all combinations as the basis for a parallel algorithm to solve this problem.

   (b) Explain how tasking is appropriate for the implementation of parallel recursive algorithms.

4. Consider 5 iterations with Newton's method running on a polynomial p, starting at $z = x + y\sqrt{-1}$, for x and y both in -1 and $+1$, for some discretization Δx and Δy.

   The program computes an n-by-n matrix with the complex numbers obtained by Newton's method.

   (a) Why is this computation suitable to apply data parallelism?

   (b) Outline the kernel function in sufficient detail to compute the CGMA ratio.
   What is the parameter that determines whether the problem will be compute or memory bound?

The questions on page 2 are for the use-computer exam. Answers are due Wednesday 8 March, at 2PM.

The context of the problems on this page is the same as on the paper-and-pencil version. Solving the questions below requires the use of a computer. In your answers, specify which computer was used.

1. Suppose we have for every angle t at our current position, a function $d(t)$ which returns the distance to the closest object in that direction of the angle t. For a given $\Delta t > 0$, evaluate the function $d(t)$ at all angles from 0 to 2π, separated from each other by Δt. A convenient choice for Δt is $2\pi/n$, for some large positive integer n.

   In your implementation, let $d(t)$ return a random double in $[0, 1]$ for every t.

   (a) Define the manager/worker interaction with message passing in C, with mpi4py or MPI.jl.

   (b) Examine the scalability for $p = 2, 4, 8, 16, 32$. Report wall clock times for each run. Justify your choice of the size of the work load so the problem scales well.

2. Given are two vectors $\mathbf{x}$ and $\mathbf{y}$, both of length n, where $y_k = f(x_k)$ for $k = 1, 2, \ldots, n$, for some function f evaluated at distinct points $x_i \neq x_j$ for $i \neq j$.

   The vector of divided differences is computed by the code below:

```
for i from 2 to n do
    for j from 1 to i-1 do
        y[i] = (y[j] - y[i])/(x[j] - x[i])
```

   (a) Use pthreads, OpenMP, or Julia to define a multithreaded implementation.

   (b) How large should n be before speedups are observed?

3. Given are n weights w_1, w_2, ... , w_n (given as positive doubles) and two doubles L and M, with $L < M$.

   The knapsack problem asks for all sums S of subsets of the weights with $L \leq S \leq M$.

   Each solution is represented as a set of indices, a subset of $\{1, 2, \ldots, n\}$.

   (a) Use OpenMP or Julia to apply tasking.
   Generate random doubles with good values for L and M.
   Show the correctness with outputs of modest size.

   (b) When timing the run times, do not take the printing of the solutions into account when computing the speedup factors.

4. Consider 5 iterations with Newton's method running on a polynomial p, starting at $z = x + y\sqrt{-1}$, for x and y both in -1 and $+1$, for some discretization Δx and Δy.

   The program computes an n-by-n matrix with the complex numbers obtained by Newton's method.

   Answer the following questions, in C++, PyCUDA, or with Julia.

   (a) Define the kernel to compute the matrix.

   (b) How large should the input data be to fully occupy the GPU?