Homework 0: due by 11:59pm on Tuesday, September 11, 2012 (Total: 100 points)

Instructor: Vivek Sarkar
Co-Instructor: Ran Libeskind-Hadas

All homeworks should be submitted using the submission system at http://cs.hmc.edu/submit. In case of problems using the script, please email your homework to vsarkar at cs.hmc.edu and hadas at cs.hmc.edu.

Honor Code Policy: All submitted homeworks are expected to be the result of your individual effort. You are free to discuss course material and approaches to problems with your other classmates, the teaching assistants and the professor, but you should never misrepresent someone elses work as your own. If you use any material from external sources, you must provide proper attribution.

1 Written Assignments (50 points total)

Please submit your solutions to the written assignments in a single text/pdf file.

1.1 Analyzing Finish-Async Programs (20 points)

Consider the following HJ code fragment:

```
finish {
  async S1;
  finish {
    async S2;
    S3;
  }
  S4;
}
S5;
```

Answer the following questions as true or false:

(a) Can S2 potentially execute in parallel with S3?

(b) Can S2 potentially execute in parallel with S4?

(c) Can S1 potentially execute in parallel with S3?

(d) Can S1 potentially execute in parallel with S4?

(e) Can S1 potentially execute in parallel with S5?

1.2 Finish Synchronization #1 (15 points)

Consider the sequential and incorrect parallel versions of the HJ code fragment included below. *Your task is to only insert finish statements in the incorrect parallel version so as to make it correct i.e., to ensure that the parallel version computes the same result as the sequential version, while maximizing the potential parallelism.*

```
// SEQUENTIAL VERSION:
  for ( p = first; p != null; p = p.next) p.x = p.y + p.z;
```

```
  for ( p = first; p != null; p = p.next) sum += p.x;

// INCORRECT PARALLEL VERSION:
  for ( p = first; p != null; p = p.next) async p.x = p.y + p.z;
  for ( p = first; p != null; p = p.next) sum += p.x;
```

1.3 Finish Synchronization #2 (15 points)

Again, consider the sequential and incorrect parallel versions of the HJ code fragment included below. Assume rows in the two-dimensional arrays are distinct objects (subarrays). *Your task is to only insert finish statements in the incorrect parallel version so as to make it correct i.e., to ensure that the parallel version computes the same result as the sequential version, while maximizing the potential parallelism.*

```
// SEQUENTIAL VERSION:
 for (int i = 0 ; i < n ; i++)
    for (int j = 0 ; j < n ; j++)
      c[i][j] = 0;
  for (int i = 0 ; i < n ; i++)
    for (int j = 0 ; j < n ; j++)
      for (int k = 0 ; k < n ; k++)
        c[i][j] += a[i][k] * b[k][j];
  System.out.println(c[0][0]);

// INCORRECT PARALLEL VERSION:
  for (int i = 0 ; i < n ; i++)
    for (int j = 0 ; j < n ; j++)
      async c[i][j] = 0;
  for (int i = 0 ; i < n ; i++)
    for (int j = 0 ; j < n ; j++)
      async
        for (int k = 0 ; k < n ; k++)
          c[i][j] += a[i][k] * b[k][j];
  System.out.println(c[0][0]);
```

2 Programming Assignments (50 points total)

2.1 Habanero-Java Setup (10 points)

All programming assignments in this course will use the Habanero Java (HJ) software. If you prefer, you can also install a release of DrHJ and HJ on your own MacOS/Linux (but not Windows) computer from the HJ Download and Setup page, but staff support for individual installations will be limited. It is instead recommended that you use the HJ software installations on HMC CS machines as follows:

1. **DrHJ IDE:** DrHJ is a pedagogic IDE for HJ that is derived from DrJava. You can use DrHJ to edit, compile and run HJ programs on whichever machine DrHJ is launched on. DrHJ works by launching one JVM for the IDE and another JVM for the program being executed. This will not pose a problem for early homeworks, but the command line version will need to be used later in the course when it will be important to devote the full computer to just the program (without perturbation from the DrHJ IDE.)

   To use the DrHJ IDE on an HMC CS lab machine, navigate to /cs/cs181e/hj in your Finder window and double click drhj.jar, or type "`java -jar /cs/cs181e/hj/drhj.jar`" in a terminal window. You

can click on Preferences to select different compile-time and run-time options that you will learn later in the course.

2. **Command line:** To use the command line version on an HMC CS machine (including the new 64-core knuth.cs.hmc.edu server), type "`source /cs/cs181e/hjsetup.txt`" in a command line window, which sets some environment variables and updates your path. Then, you should be able to run the `hjc` and `hj` commands for compilation and execution of HJ programs, analogous to `javac` and `java` for Java programs. You can type `hjc --help` and `hj --help` to see different compile-time and run-time options that you will learn later in the course.

To get credit for this section, download the HelloWorld.hj program, compile and run it, and include the generated output in your homework submission.

If you are unfamiliar with DrJava, here are some instructions to compile and run HJ programs using the DrHJ IDE:

- Launch DrHJ as described earlier.
- Open the HJ program, click on the Open button in the top panel, navigate to the folder containing HelloWorld.hj, select the file, and click Open.
- Compile the HJ program by clicking on the Compile button in the top panel. The "Compiler Output" tab in the bottom panel should show "Compilation Completed". (Always check the "Console" tab in the bottom panel to see if there were any errors during compilation shown in red.)
- Go to the "Interactions" tab in the bottom panel. Run the program by typing "*run HelloWorld*"

2.2 Abstract Performance Metrics (20 points)

Download the Search.hj program. This sequential program performs a brute-force search for a pattern string in a text string. This program has been instrumented to count each character comparison as 1 unit of work from the viewpoint of abstract performance metrics, and ignore everything else.

To execute an HJ program with an option to generate abstract performance metrics, select "Show Abstract Execution Metrics" in DrHJ's Compiler Option preferences, or (if youre not using DrHJ) type the following command on the command line, "`hj -perf=true` ..."

Your lab assignment is to convert the sequential program to a parallel program that produces the correct answer with a smaller critical path length (ideal parallel time) than the sequential version. Include the parallel program in your submissing, along with ideal parallel times for the sequential and parallel versions.

2.3 Data Race Detection and Correction (20 points)

Download the ReciprocalArraySum.hj program. This program has two methods, `seqArraySum()` and `parArraySum()` , that contain sequential and parallel implementations of the same computation. However, the parallel version has a bug — specifically, a data race.

Your task is as follows:

1. *Use the* `-racedet` *command-line option or the data race detection preference in DrHJ to enable dynamic data race detection when compiling your HJ program. It is recommended that you use a small input size (e.g., 10) when executing this compiled version. Include the output from data race detection in your homework submission. Note that the line:column coordinates pinpoint the locations of the memory accesses participating in the data race.*
2. *Fix the data race, and submit a corrected version of the HJ program.*
3. *Include timings of the sequential and parallel methods for input sizes 10, 100, 1000, 10000, 100000, 1000000, and 10000000. Discuss when the sequential or parallel version is faster and why.*