Drug Design

In the folder `~/CSinParallel/RaspberryPiBasics/drugdesign/` you will find three implementations of a program that attempts to locate small molecules called *ligands* that are good candidates for use as drugs. When a protein associated with a disease of interest is identified, its three-dimensional structure is found either experimentally or through a molecular modeling computation. A collection of ligands is tested against the protein. A score is set based on these binding properties, and the best scores are flagged, identifying ligands that would make good drug candidates. For simplicity, the programs we implement use strings to represent proteins and ligands. The three parallel implementations can be summarized as follows:

- `drugdesign-static`: Uses OpenMP to assign chunks of ligands at compile time to cores.
- `drugdesign-dynamic`: Uses OpenMP to assign ligands to cores dynamically (at run-time).
- `drugdesign-threads`: Uses C++11 threads (instead of OpenMP threads) to solve the problem.

To compile each version, cd into each individual directory and type make. Your goal is to determine which program is the fastest.

Measure Run-Time

To measure the running time of each implementation, use `time –p`. For example:

```
> time  -p  ./drugdesign-static  1
```

Runs the drugdesign-dynamic executable on a single thread. Fill out the table below (use `real` for time):

Time (s)	1 Thread	2 Threads	3 Threads	4 Threads
drugdesign-static				
drugdesign-dynamic				
drugdesign-threads				

Compute the Speedup

Speedup measures how much faster the parallel execution of a program is compared to the serial (single threaded) execution. The Speedup of a *n* core approach (S_n) is computed by the following equation:

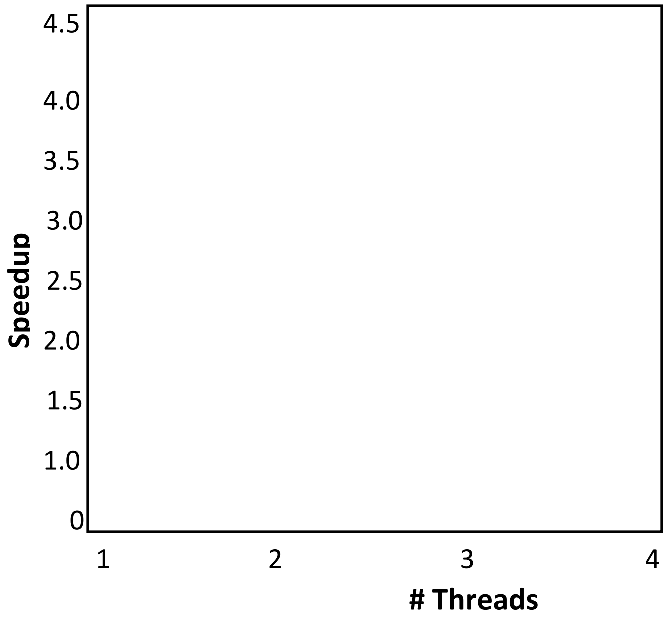
$$S_n = \frac{Time(1\ thread)}{Time\ (n\ threads)}$$

Hint: Consider using the calculator application on the Raspberry Pi 2 to help you with the calculation!

Speedup	2 Threads	3 Threads	4 Threads
drugdesign-static			
drugdesign-dynamic			
drugdesign-threads			

Plot Your Results

Problem 2c: Plot your speedup results in the area below. Use different lines for each of the different applications.



Discussion Questions

1. Which approach is the fastest?

2. Determine the number of lines in each file (use `wc -l`). How does the C++11 implementation compare to the OpenMP implementations?

3. Increase the number of threads to 5 threads. What is the run time for each?

4. Increase the maximum ligand length to 7, and rerun each program. Are you speedup results similar?