```
  # Update points along line
  do i = 1, npoints
    newval(i) = (2.0 * values(i)) - oldval(i)
    + (sqtau * (values(i-1) - (2.0 * values(i)) + values(i+1)))
  end do
end do

#Collect results and write to file
if I am LEADER
  receive results from each WORKER
  write results to file
else if I am WORKER
  send results to LEADER
endif
```

Your task: Divide the amplitude array into t partitions (where t is the number of tasks). Note that in this case, there are data dependencies between tasks (between partition edges). Interior elements of each partition are independent of other tasks, but at partition boundaries, each partition's results depends upon results generated by another partition's task. You will have to synchronize task execution and share results at these edges. You can just print the numeric results, but a dynamic display plotting the string over time would be much more engaging.

Possible parallel programming environments (in no particular order):

1) OMP (works out of the box with Visual Studio and gcc C and C++)
2) RISC-V atomic extensions with C or C++ (you will have to wrap with inline assembly as shown in class)
3) Python multiprocessing, threading, or IPython
4) C# System.Threading package.
5) MPJExpress (http://mpj-express.org/), or Java Threads for Java
6) Some other parallel prog. environment that catches your eye (there are a bunch listed here): https://www.cise.ufl.edu/research/ParallelPatterns/PatternLanguage/Background/ProgEnvs.htm

Option 2: Cache Simulator

Note: There are many cache simulators available on the Internet. Your goal is NOT to employ one of them, rather it is to develop a functional simulator of you own making.

For this project you will create data cache simulator. The simulator should be configurable (configuration can be hard-coded in the program, or obtained from the user) to support:

- An N-way associative cache, where N is power of 2 from 0 to 16;
- LRU, Random, and NMRU+Random replacement policies;
- Cache size up to 256KB;
- Block size up to 8 words.

Assume a 32-bit architecture.

CSCI 341 Fall 2022 Lab 6

The input to your simulator will be an address trace file (`tracefile.zip`) found under the Resources tab on Piazza, or using this direct link:

https://piazza.com/class_profile/get_resource/l4q346ccgoz38w/la2zps7il8569m

This file contains the trace of access instructions executed for some program. Be sure to unzip before use. Each line represents a byte address referenced by an executed load or store instruction. For this assignment, all memory accesses are considered the same (i.e., writes are handled no differently than reads).

The output of running your cache simulator should look something like:

Cache_Size: 32 KBytes
Block_Size: 4 words (16 bytes)
Associativity: 2 way
Replacement_Policy: NMRU+Random
Total_Number_of_Accesses: 1500000
Cache_Hits: 1005939
Cache_Misses: 494061
Cache_Hit_Rate: 0.67
Cache_Miss_Rate:0.33

Exercise your simulator for a number of plausible sets of values, testing both end conditions and likely choices. Your goal is to find best design choice (simplest design with good-to-excellent performance), configuration choices for cache sizes of 16, 32, 64, 128, and 256KB for this data set, and to plot these results.

Option 3: An approved individual adventure of your choosing.

Make a written proposal to your section instructor submitted by email no later than November 10 @ 10:00PM (sooner is better). You will get feedback no later than 5:00 PM, Nov. 12th indicating whether your project is approved and what changes (if any) are required to your proposed scope of work and deliverables.

Proposals must include the following:

1. **Your Name**
2. **Project Summary – one or two paragraphs describing the project and why it interests you.**
3. **Proposed Scope of Work – exactly what you propose to accomplish, and how you will evaluate your work; be specific**
4. **Deliverables – what you will turn in by the due date; be specific**
5. **Timeline – a description of project milestones, and an estimate of how long each will take**
6. **Other Information (optional) – anything else you want to include**