Project #4

Functional Decomposition

100 Points

Due: May 16

Introduction

This project will use parallelism, not for speeding data computation, but for programming convenience. You will create a month-by-month simulation in which each agent of the simulation will execute in its own thread where it just has to look at the state of the world around it and react to it.

You will also get to exercise your creativity by adding an additional "agent" to the simulation, one that impacts the state of the other agents and is impacted by them.

Requirements

1. You are creating a month-by-month simulation of a grain-growing operation. The amount the grain grows is affected by the temperature, amount of precipitation, and the number of "graindeer" around to eat it. The number of graindeer depends on the amount of grain available to eat.
2. The "state" of the system consists of the following global variables:

```
int       NowYear;               // 2017 - 2022
int       NowMonth;              // 0 - 11

float     NowPrecip;             // inches of rain per month
float     NowTemp;               // temperature this month
float     NowHeight;             // grain height in inches
int       NowNumDeer;            // number of deer in the current
population
```

3. Your basic time step will be one month. Interesting parameters that you need are:

```
const float GRAIN_GROWS_PER_MONTH =             8.0;
const float ONE_DEER_EATS_PER_MONTH =           0.5;

const float AVG_PRECIP_PER_MONTH =              6.0;    // average
const float AMP_PRECIP_PER_MONTH =              6.0;    // plus or minus
const float RANDOM_PRECIP =                        2.0;    // plus or minus
noise

const float AVG_TEMP =                             50.0;   // average
const float AMP_TEMP =                             20.0;   // plus or minus
const float RANDOM_TEMP =                          10.0;   // plus or minus
noise

const float MIDTEMP =                              40.0;
const float MIDPRECIP =                            10.0;
```

Units of grain growth are inches.
Units of temperature are degrees Fahrenheit (°F).
Units of precipitation are inches.

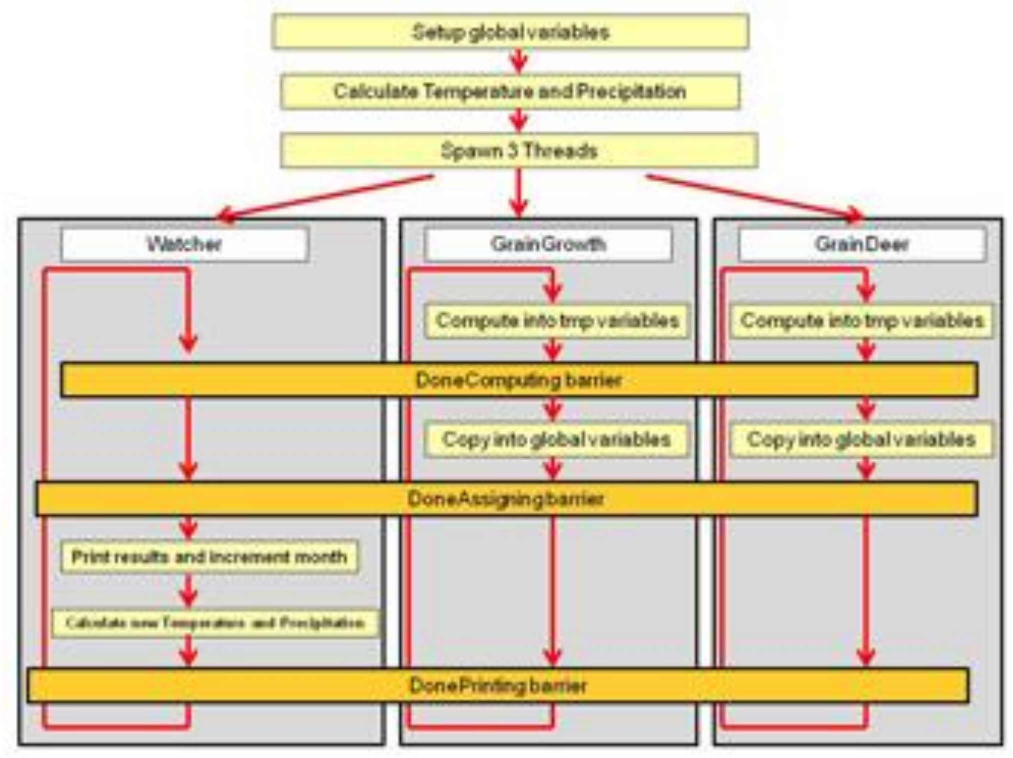
4. Because you know ahead of time how many threads you will need (3 or 4), start the threads with a `parallel sections` directive.
5. The temperature and precipitation are a function of the particular month.

   To keep this simple, a year consists of 12 months of 30 days each. The first day of winter is considered to be January 1. As you can see, the temperature and precipitation follow cosine and sine wave patterns with some randomness added.

6. **In addition to this, you must add in some other phenomenon that directly or indirectly controls the growth of the grain and/or the graindeer population.** Your choice of this is up to you.
7. You are free to tweak the constants to make everything turn out "more interesting".

Use of Threads

As shown here, you will spawn three threads (four, when you add your own agent):


Setup global variables
Calculate Temperature and Precipitation
Spawn 3 Threads
Watcher
GrainGrowth
GrainDeer
Compute into tmp variables
Compute into tmp variables
DoneComputing barrier
Copy into global variables
Copy into global variables
DoneAssigning barrier
Print results and increment month
Calculate new Temperature and Precipitation
DonePrinting barrier

The **GrainGrowth** and **GrainDeer** threads will each compute the next grain height and the next number of deer based on the current set of global state variables. They will compute these into local, temporary, variables. They both then will hit the DoneComputing barrier.

At that point, both of those threads are done computing using the current set of global state variables. Each thread should then copy the local variable into the global version. All 3 threads will then hit the **DoneAssigning** barrier.

At this point, the Watcher thread will print the current set of global state variables, increment the month count, and then use the new month to compute the new Temperature and Precipitation. Note that the **GrainGrowth** and **GrainDeer** threads can't proceed because there is a chance they would re-compute the global state variables before they are done being printed. All 3 threads will then hit the **DonePrinting** barrier.

After spawning the threads, the main program should wait for the **parallel sections** to finish.

Each thread should return when the year hits 2023 (giving us 6 years, or 72 months, of simulation).

Remember that this description is for the core part of the project, before you add your own agent to the simulation. That will involve another thread and some additional interaction among the global state variables.

Quantity Interactions

The Carrying Capacity of the graindeer is the number of inches of height of the grain. If the number of graindeer exceeds this value at the end of a month, decrease the number of graindeer by one. If the number of graindeer is less than this value at the end of a month, increase the number of graindeer by one.

Each month you will need to figure out how much the grain grows. If conditions are good, it will grow by `GRAIN_GROWS_PER_MONTH`. If conditions are not good, it won't.

You know how good conditions are by seeing how close you are to an ideal temperature (°F) and precipitation (inches). Do this by computing a Temperature Factor and a Precipitation Factor like this:

$$TF = e^{-\left(\frac{T - MidTemp}{10}\right)^2}$$

$$PF = e^{-\left(\frac{P - MidPrecip}{10}\right)^2}$$

Results

Turn in your code and your PDF writeup. Your writeup will consist of:

1. What your own-choice quantity was and how it fits into the simulation.
2. A table showing values for temperature, precipitation, number of graindeer, height of the grain, and your own-choice quantity as a function of month number.
3. A graph showing temperature, precipitation, number of graindeer, height of the grain, and your own-choice quantity as a function of month number.
4. A commentary about the patterns in the graph and why they turned out that way. What evidence in the curves proves that your own quantity is actually affecting the simulation?