5/7/19
Project 3: Functional Decomposition

Background: Ran program on Flip2 with up time load average of 1.98.

Results:

1. I used a flood event as a personal choice event that affects the grain-deer and grain height. I included two instances of a flooding event, occurring in months 40 and 70 denoted by the number 25, both of which increase the grain height by 2 inches and increase the grain-deer population by one because it seems practical that grain crops would grow better with atypical events of increased rain. Only two floods were included because several different test cases (more drastic deer count and grain height alterations and flooding frequencies) resulted in many deer count of zero populations, making this simulation nonviable. This choice to include two flood events worked well to observe a change in the included graph further down.

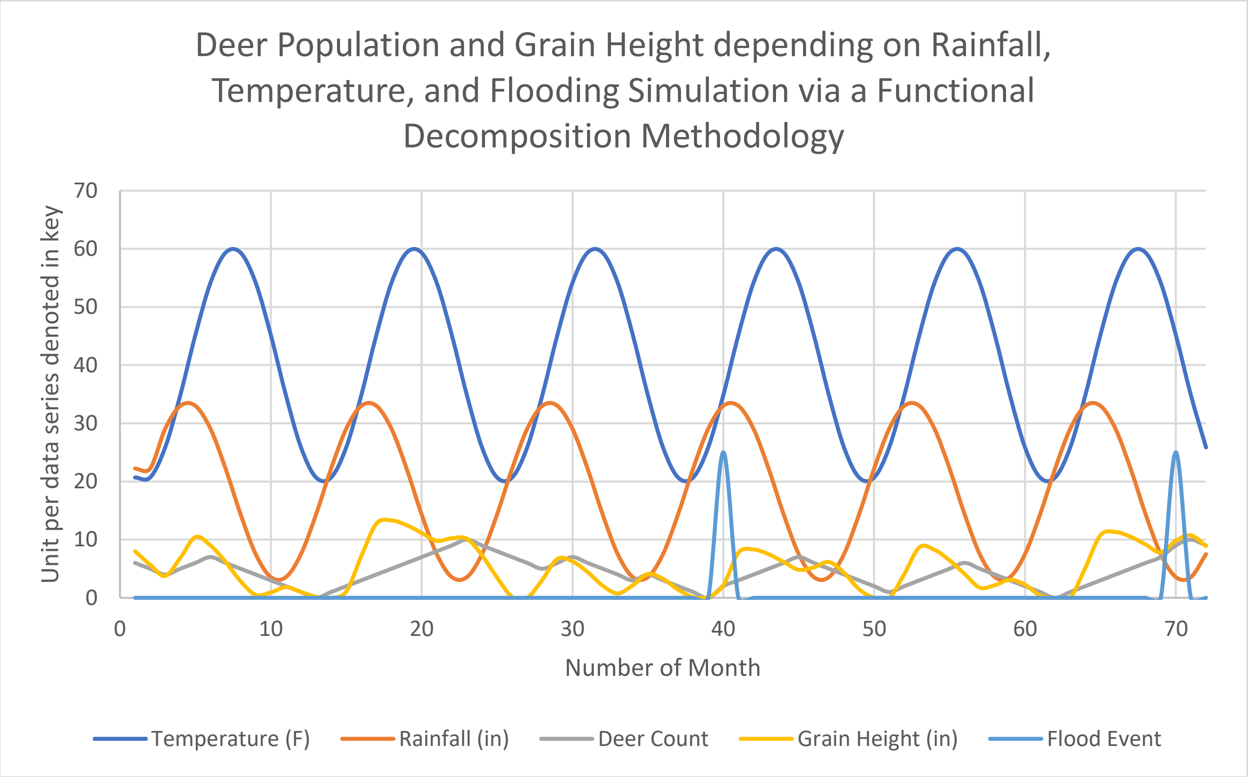
2. Data table showing the Month Number, Temperature in Fahrenheit, Grain-Deer population, Grain-Height in inches, and when a flood event occurs as denoted by the number 25.

month	temp ©	Temperature (F)	Rainfall (in)	Deer Count	Grain Height (cm)	Grain Height (in)	Flood Event
1	-6.28283	20.69091	22.2275	6	20.32	8	0
2	-6.28283	20.69091	22.2275	5	14.4508	5.689291	0
3	-3.40706	25.86729	29.0594	4	9.53146	3.752543	0
4	1.57392	34.83306	33.0038	5	17.4066	6.852992	0
5	7.32545	45.18581	33.0038	6	26.5241	10.44256	0
6	12.3064	54.15152	29.0594	7	22.8604	9.000157	0
7	15.1822	59.32796	22.2275	6	15.7177	6.188071	0
8	15.1822	59.32796	14.3387	5	7.2287	2.845945	0
9	12.3064	54.15152	7.50681	4	1.27846	0.503331	0
10	7.32545	45.18581	3.56241	3	2.34352	0.922646	0
11	1.57392	34.83306	3.56241	2	4.69307	1.847665	0
12	-3.40706	25.86729	7.50681	1	2.56176	1.008567	0
13	-6.28283	20.69091	14.3387	0	0.425721	0.167607	0
14	-6.28283	20.69091	22.2275	1	0	0	0
15	-3.40706	25.86729	29.0594	2	2.70071	1.063272	0
16	1.57392	34.83306	33.0038	3	18.1959	7.16374	0
17	7.32545	45.18581	33.0038	4	32.3933	12.75327	0
18	12.3064	54.15152	29.0594	5	33.8097	13.31091	0
19	15.1822	59.32796	22.2275	6	31.7469	12.49878	0
20	15.1822	59.32796	14.3387	7	28.3379	11.15665	0
21	12.3064	54.15152	7.50681	8	24.9277	9.814055	0
22	7.32545	45.18581	3.56241	9	25.9928	10.23339	0
23	1.57392	34.83306	3.56241	10	25.8023	10.15839	0
24	-3.40706	25.86729	7.50681	9	18.591	7.319291	0

25	-6.28283	20.69091	14.3387	8	8.83497	3.478335	0
26	-6.28283	20.69091	22.2275	7	0.425721	0.167607	0
27	-3.40706	25.86729	29.0594	6	0	0	0
28	1.57392	34.83306	33.0038	5	7.87518	3.100465	0
29	7.32545	45.18581	33.0038	6	16.9926	6.69	0
30	12.3064	54.15152	29.0594	7	15.8689	6.247598	0
31	15.1822	59.32796	22.2275	6	11.2662	4.435512	0
32	15.1822	59.32796	14.3387	5	5.31723	2.093398	0
33	12.3064	54.15152	7.50681	4	1.90699	0.750783	0
34	7.32545	45.18581	3.56241	3	5.51205	2.170098	0
35	1.57392	34.83306	3.56241	4	10.4016	4.095118	0
36	-3.40706	25.86729	7.50681	3	8.2703	3.256024	0
37	-6.28283	20.69091	14.3387	2	3.59426	1.415063	0
38	-6.28283	20.69091	22.2275	1	0.265009	0.104334	0
39	-3.40706	25.86729	29.0594	0	0.42572	0.167606	0
40	1.57392	34.83306	33.0038	2	5.50572	2.167606	25
41	7.32545	45.18581	33.0038	3	19.7031	7.757126	0
42	12.3064	54.15152	29.0594	4	21.1195	8.314764	0
43	15.1822	59.32796	22.2275	5	19.0568	7.502677	0
44	15.1822	59.32796	14.3387	6	15.6478	6.160551	0
45	12.3064	54.15152	7.50681	7	12.2375	4.817913	0
46	7.32545	45.18581	3.56241	6	13.3026	5.237244	0
47	1.57392	34.83306	3.56241	5	15.6522	6.162283	0
48	-3.40706	25.86729	7.50681	4	10.9808	4.32315	0
49	-6.28283	20.69091	14.3387	3	3.7648	1.482205	0
50	-6.28283	20.69091	22.2275	2	0	0	0
51	-3.40706	25.86729	29.0594	1	0	0	0
52	1.57392	34.83306	33.0038	2	10.4152	4.100472	0
53	7.32545	45.18581	33.0038	3	22.0726	8.69	0
54	12.3064	54.15152	29.0594	4	20.9489	8.247598	0
55	15.1822	59.32796	22.2275	5	16.3462	6.435512	0
56	15.1822	59.32796	14.3387	6	10.3972	4.093386	0
57	12.3064	54.15152	7.50681	5	4.44699	1.750783	0
58	7.32545	45.18581	3.56241	4	5.51206	2.170102	0
59	1.57392	34.83306	3.56241	3	7.86161	3.095122	0
60	-3.40706	25.86729	7.50681	2	5.7303	2.256024	0
61	-6.28283	20.69091	14.3387	1	1.05426	0.415063	0
62	-6.28283	20.69091	22.2275	0	0	0	0
63	-3.40706	25.86729	29.0594	1	0.160711	0.063272	0
64	1.57392	34.83306	33.0038	2	13.1159	5.16374	0
65	7.32545	45.18581	33.0038	3	27.3133	10.75327	0
66	12.3064	54.15152	29.0594	4	28.7297	11.31091	0
67	15.1822	59.32796	22.2275	5	26.6669	10.49878	0
68	15.1822	59.32796	14.3387	6	23.2579	9.156654	0

69	12.3064	54.15152	7.50681	7	19.8477	7.814055	0
70	7.32545	45.18581	3.56241	9	24.9277	9.814055	25
71	1.57392	34.83306	3.56241	10	27.2773	10.73909	0
72	-3.40706	25.86729	7.50681	9	22.6059	8.899961	0

3. Graph



4. Graph discussion: In the graph as provided in the program layout, the temperature follows a cosine function pattern while the rainfall follows a sinusoidal function pattern. There appears to be a positive correlation of the deer count and grain height where both series of data follow an identical pattern, yet the deer population appears to move at a slightly lower rate of growth than the grain height change and that makes sense when considering the grain height changes by a larger value (can be multiple inches at a time) whereas the deer population growth is set to change one at a time either up one or down one. The two flood events as denoted by the lighter blue line on months 40 and 70 show a spike in the deer population that actually goes beyond the rate of growth (slope) from the grain height in month 40 and the deer population growth matches an identical grain height growth (slope) for month 70 and this is because for both flood events the deer population is increased by one and for both days because the deer population was below the grain height the deer population then increased by two in both floods.

```
#include <iostream>
#include <omp.h>
#include <math.h>
#include <stdlib.h>

//fx prototypes
void GrainDeer();
void Grain();
void Watcher();
void floods();
void computeWeather();
void getData();
float SQR(float);
float Ranf(unsigned int*, float, float);
int Randf(unsigned int*, int, int);
unsigned int seed = 0;  // a thread-private variable

// The "state" of the system consists of the following global variables.
int    NowYear;                    // 2019 - 2024
int    NowMonth;           // 0 - 11
//used to print out month count for convenient graphing
int monthCounter;

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
int   flooding = 0;

// Initialize temperature and precipitation constants.
const float GRAIN_GROWS_PER_MONTH =             8.0;   // inches
const float ONE_DEER_EATS_PER_MONTH =     0.5;

const float AVG_PRECIP_PER_MONTH =       6.0;   // average (inches)
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 (deg_F)
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;

int main()
{
    // Starting state (feel free to change):
       NowNumDeer = 5;
       NowHeight =  8;
    // starting date and time:
    NowYear = 2019;
       NowMonth =    0;
    monthCounter=0;

       computeWeather(); //trials
       getData();
       omp_set_num_threads( 4 );   // same as # of sections
       #pragma omp parallel sections
```

```
        {
                #pragma omp section
                {
                        GrainDeer( );
                }

                #pragma omp section
                {
                        Grain( );
                }

                #pragma omp section
                {
                        Watcher( );
                }

                #pragma omp section
                {
                        floods( );
                }
        }       // implied barrier
return 0;
}

//graindeer sim.
void GrainDeer()
{
        while( NowYear < 2025 )
        {
                // compute a temporary next-value for this quantity
                // based on the current state of the simulation:
                float tempD = NowNumDeer;
                if(tempD > NowHeight)
                        tempD--;
                else
                        tempD++;

                // DoneComputing barrier:
                #pragma omp barrier
                if(flooding == 0)
                {
                        NowNumDeer = tempD;
                }
                // DoneAssigning barrier:
                #pragma omp barrier
                // DonePrinting barrier:
                #pragma omp barrier
        }

}

//grain growth sim
void Grain()
{
                while( NowYear < 2025 )
        {
                // compute a temporary next-value for this quantity
                // based on the current state of the simulation:
```

```
              float tempH = NowHeight;
        float heighTemp = tempH;
        //determine growth based on rain and temp
        float precipFactor = exp(   -SQR(  ( NowPrecip - MIDPRECIP ) / 10.  )   );
              float tempFactor = exp(   -SQR(  ( NowTemp - MIDTEMP ) / 10.  )   );
              //get growth
              tempH += tempFactor * precipFactor * GRAIN_GROWS_PER_MONTH;
              heighTemp -= (float)NowNumDeer * ONE_DEER_EATS_PER_MONTH;
              if(heighTemp > 0)
            tempH = heighTemp;
              else
                     tempH = 0;

              // DoneComputing barrier:
              #pragma omp barrier
              if(flooding == 0)
                     NowHeight = tempH;
              // DoneAssigning barrier:
              #pragma omp barrier
              // DonePrinting barrier:
              #pragma omp barrier
       }
}

void Watcher()
{
              while( NowYear < 2025 )
       {
              // computer and assigned barriers
              #pragma omp barrier
              #pragma omp barrier
              //display global vars stat
              getData();
              // Increment month count.
              NowMonth++;
              monthCounter++;
              if(NowMonth == 12){
            NowMonth = 0;
            NowYear++; }
              //get new temp and rain
              computeWeather();
              // DonePrinting barrier:
              #pragma omp barrier
       }
}
// floods causing grain growth and deer death
void floods()
{

       while( NowYear < 2025 )
       {

              // compute a temporary next-value for this quantity
              // based on the current state of the simulation:
              float tempD = NowNumDeer;
              float tempH = NowHeight;
              flooding = 0;
              if(NowYear == 2022 && NowMonth == 2)
              {
            flooding = 25;   //set flooding to active state
                     //flood lets grain grow 2 more inches
                     tempH = (tempH+2);
                     //flood causes 1 deer to die
                     tempD= (tempD+1);
              }
        if(NowYear == 2023 && NowMonth == 8)
              {
            flooding = 25;   //set flooding to active state
                     //flood lets grain grow 2 more inches
                     tempH = (tempH+2);
                     //flood causes 1 deer to die
                     tempD= (tempD+1);
              }

              // DoneComputing barrier:
              #pragma omp barrier
              if(flooding == 25)
              {
                     NowNumDeer = tempD;
                     NowHeight = tempH;
              }

              // DoneAssigning barrier:
              #pragma omp barrier

              // DonePrinting barrier:
              #pragma omp barrier
              //. . .
       }

}
// Print current set of global state variables.
void getData()
{
 //   ofstream data;
 //   data.open("dataFile.txt");
 //   data << monthCounter << " " <<  (5./9.)*(NowTemp-32) << " " << NowPrecip*2.54 << "
" << NowNumDeer << " " << NowHeight*2.54 << " " << flooding << "\n";
       std::cout << monthCounter << " " <<  NowTemp << " " << NowPrecip << " " <<
NowNumDeer << " " << NowHeight << " " << flooding << "\n";
 //   data.close();
}

//monthly temp and rain calculation
void computeWeather()
{

       float ang = (  30.*(float)NowMonth + 15.  ) * ( M_PI / 180. );

       float temp = AVG_TEMP - AMP_TEMP * cos( ang );
       unsigned int seed = 0;
       NowTemp = temp + Ranf( &seed, -RANDOM_TEMP, RANDOM_TEMP );

       float precip = AVG_PRECIP_PER_MONTH + AMP_PRECIP_PER_MONTH * sin( ang );
       NowPrecip = precip + Ranf( &seed,  -RANDOM_PRECIP, RANDOM_PRECIP );
       if( NowPrecip < 0. )
       {
              NowPrecip = 0.;
       }

}

//square fx
float SQR(float x)
{
       return x*x;
}

//rand num seeds
float
Ranf( unsigned int *seedp,  float low, float high )
{
        float r = (float) rand_r( seedp );              // 0 - RAND_MAX

        return(   low  +  r * ( high - low ) / (float)RAND_MAX   );
}

int
Ranf( unsigned int *seedp, int ilow, int ihigh )
{
        float low = (float)ilow;
        float high = (float)ihigh + 0.9999f;

        return (int)(  Ranf(seedp, low,high) );
}
```