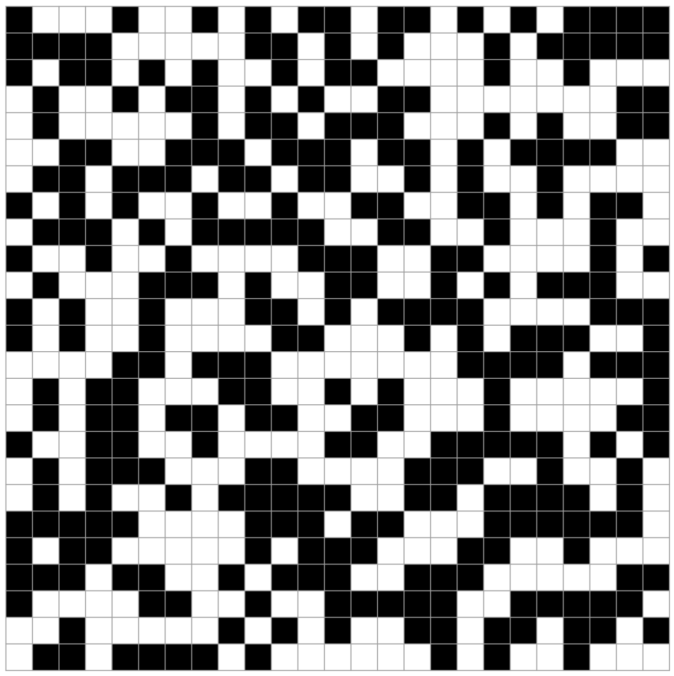
Simulation of the 2D Ising Model

Background & Setup

The idea of this program is to simulate, on a small scale, the internal structure and dynamics of a ferromagnet or and antiferromagnet. Below are snipets of *Mathematica* code that will be assembled into a working program that you will use to explore the 2D Ising model.

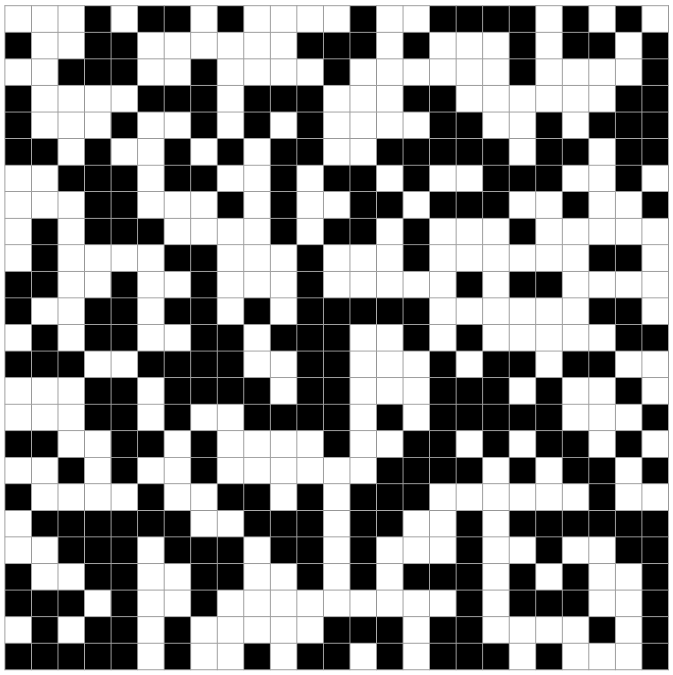
```
ClearAll["Global`*"]

(*specifies the lattice size*)

n = 25;

(*generates a random inital setup*)

SeedRandom[1]; initLat = 2 Table[Random[Integer], {n}, {n}] - 1;

(*thie creates a pictoral plot of the spin structure with
  white representing spin-up and black representing spin-down*)

ArrayPlot[initLat, ColorRules → {1 → White, -1 → Black}, Mesh → All]
```



```
(*sets the initial values for the Heisenberg Hamiltonian,
external field and coupling constant.  If J>0,
then the system is a ferromagnet; if J<0, the system is an antiferromagnet*)
```

```
B = 0; J = 0.01;

(*a table consisting of all possible energy changes if a single site is flipped
 based on all possible configurations of the sites 4 nearest neighbors*)

δE[ 1, 4] =  2 (B + 4 J);
δE[ 1, 2] =  2 (B + 2 J);
δE[ 1, 0] =  2 (B + 0 J);
δE[ 1, -2] =  2 (B - 2 J);
δE[ 1, -4] =  2 (B - 4 J);
δE[-1, 4] = -2 (B + 4 J);
δE[-1, 2] = -2 (B + 2 J);
δE[-1, 0] = -2 (B + 0 J);
δE[-1, -2] = -2 (B - 2 J);
δE[-1, -4] = -2 (B - 4 J);

(*Metropolis algorithm used to randomly select a site, test the energy,
and decide to flip the spin or not.  Has periodic boundaries.*)

flipStep =
  (
    L = #; (*the # symbol is a literal
     call to the array that this function will work on*)

    {i1, i2} = {RandomInteger[{1, n}], RandomInteger[{1, n}]}; (*selects a site*)

    (*the following 4 lines create statements to isolate
     the 4 neighest neighbors to the chosen site.  The If,Then,
    Else structure allows to the possiblilty that the chosen site
     sits on the edge of the grid, in which case it wraps around*)

    If[i1 == n, down = 1, down = i1 + 1];
    If[i2 == n, right = 1, right = i2 + 1];
    If[i1 == 1, up = n, up = i1 - 1];
    If[i2 == 1, left = n, left = i2 - 1];

    (*calculates the energy
     difference of flipping based on the 4 nearest neighbors*)

    eDiff = L[[down, i2]] + L[[up, i2]] + L[[i1, right]] + L[[i1, left]];

    (*this statement is the implimentation of the Metropolis alrogithm.  The
      quantifier is an OR statement.  if the energy difference δE<0,
    then the site is flipped.  If it is positive,
    the a random number is generated and compared to exponential of the
     energy difference and, if it is greater, the site is flipped*)

    If[δE[L[[i1, i2]], eDiff] < 0 || Random[] < Exp[-δE[#[[i1, i2]], eDiff]],
     L[[i1, i2]] = -L[[i1, i2]]; L, L];

    L (*this returns the new lattice configuration*)

   ) & ;(*the & symbol is required at
 the end of a literal function call like this*)

(*sets the number of iterations*)
```

```
count = 500 000;

(*builds an nested array of lattices, one for each step*)

flipList = NestList[flipStep, initLat, count];

(*gives a visualization of the last lattice in the calculation*)

ArrayPlot[flipList[[count]], ColorRules → {1 → White, -1 → Black}, Mesh → All]
```



Program

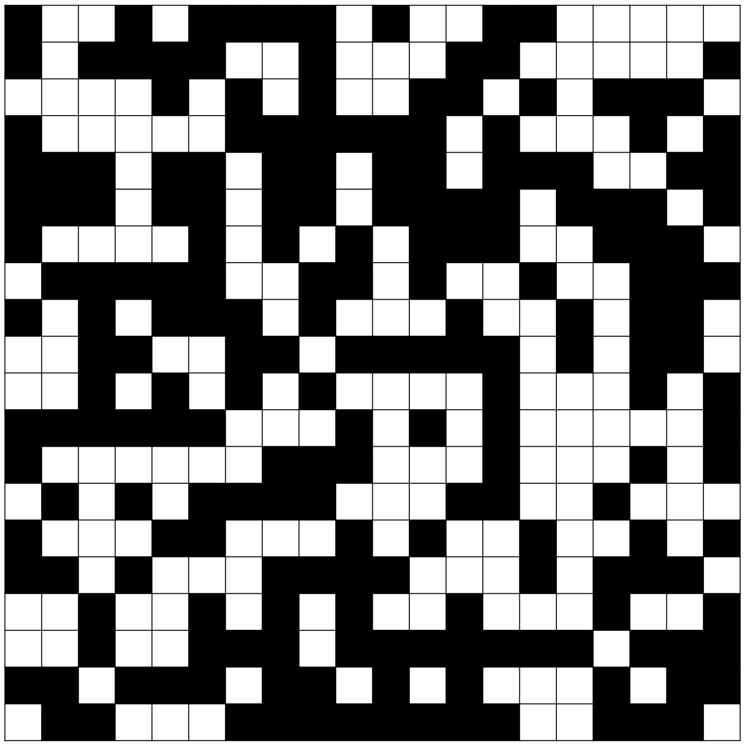
This section is the chunk of code that you will run, the Ising2D function, to answer questions about the behavior of a FM and an AFM system. You will also be required to modify it a bit in some later problems.

The variable *n* controls the dimensions of the lattice while the variable *count* controls the number of iterations. **Be warned** that this program can get memory intenstive. The larger you make *n* and *count*, the longer you can expect to hang out. Also, if you want to use the last line of code in this section, the one that animates the result, I would recommend tunring off Dynamic Upadate (do this from the menu bar at the top: Evalutaion > Dynamic Updating Disabled/Enabled). If you don’t, then everytime you run the program, *Mathematica* will try and update it as well, which gets very memory intensive.

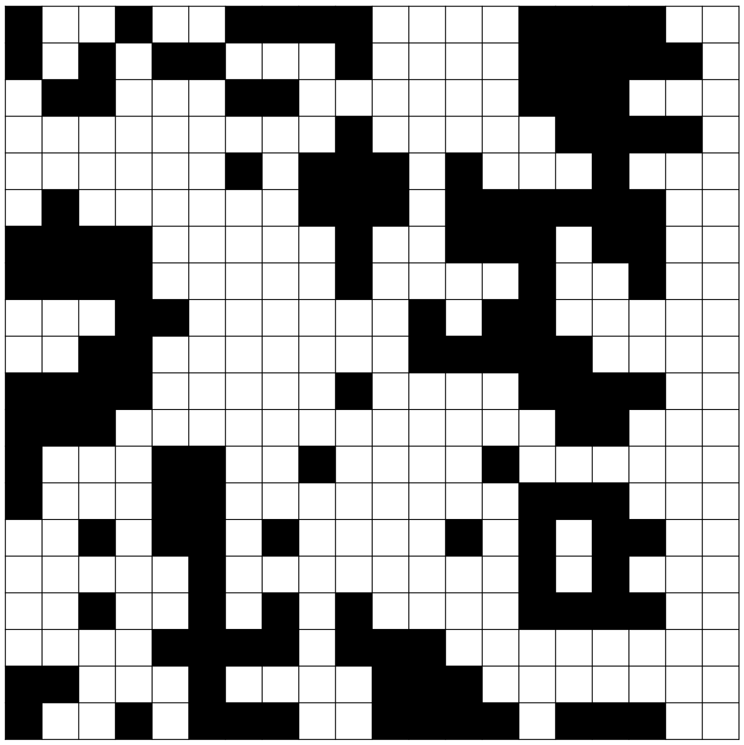
```
ClearAll["Global`*"]
```

```
Ising2D[n_, count_, B_, J_, T_] := (*in the actual program,
 temperature is implimented.  The value of the Botlzman constant, k = 1*)
 Module[{}, (*Module just groups all of the commands together*)

  initLat = 2 Table[Random[Integer], {n}, {n}] - 1;

  δE[1, 4] = 2 ((B + 4 J)/T); δE[1, 2] = 2 ((B + 2 J)/T);
  δE[1, 0] = 2 ((B + 0 J)/T); δE[1, -2] = 2 ((B - 2 J)/T);
  δE[1, -4] = 2 ((B - 4 J)/T); δE[-1, 4] = -2 ((B + 4 J)/T);
  δE[-1, 2] = -2 ((B + 2 J)/T); δE[-1, 0] = -2 ((B + 0 J)/T);
  δE[-1, -2] = -2 ((B - 2 J)/T); δE[-1, -4] = -2 ((B - 4 J)/T);

  flipStep =
   (
     L = #;

     {i1, i2} = {RandomInteger[{1, n}], RandomInteger[{1, n}]};

     If[i1 == n, down = 1, down = i1 + 1];
     If[i2 == n, right = 1, right = i2 + 1];
     If[i1 == 1, up = n, up = i1 - 1];
     If[i2 == 1, left = n, left = i2 - 1];

     eDiff = L[[down, i2]] + L[[up, i2]] + L[[i1, right]] + L[[i1, left]];

     If[δE[L[[i1, i2]], eDiff] < 0 || Random[] < Exp[-δE[#[[i1, i2]], eDiff]],
      L[[i1, i2]] = -L[[i1, i2]]; L, L];

     L

    ) &; (*end flipStep*)

  flipList = NestList[flipStep, initLat, count];

 ] (*end Module*)

nSteps = 1000;

Ising2D[20, nSteps, 0, 1, 2.5]
```

```
ArrayPlot[initLat, ColorRules → {1 → White, -1 → Black}, Mesh → All, MeshStyle → Black]
(*this is the inital lattice arrangement*)
```



```
ArrayPlot[Last[flipList], ColorRules → {1 → White, -1 → Black},
 Mesh → All, MeshStyle → Black] (*this is the final lattice arrangement*)
```



```
Animate[ArrayPlot[flipList[[i]], ColorRules → {1 → White, -1 → Black},
  Mesh → All, MeshStyle → Black], {i, 1, nSteps, 1},
 AnimationRunning → False] (*this simply animates the result,
showing the evolution of the lattice on each individual step*)
```

Questions

Problem 1

a.) Run the program with n=20 at a variety of temperatures, say T = 10, 5, 2.5, 1, and 0.1, using about 1000 steps or so. At each temperature, roughly estimate the size of the emerging clusters.

b.) Repeat part (a) with n=50. Do the cluster sizes change? Explain.

c.) Run the program with n=20 at tempertures T=2, 1, and 0.5 with a variety of step values. Estimate the average magnetization of the lattice at each temperature. Discuss the results. Do you ever see **symmetry breaking**, the point where the whole system takes on one spin value?

d.) Run the program with n=10 at T = 2.5 using at least 100,000 steps. Using the animation code, watch the system evolve. Discuss and explain this behavior.

e.) Based on the previous parts, estimate where the critical temperature of the system is. Try using larger lattices with more iterations (as much as your computer can handle) to get a good feeling for what is happening.

Problem 2

Modify the provided function to calculate the average energy of the lattice (energy over all iterations). Then run the program with n=20 for temperatures from T=5 to T=1 (using an adequate number of steps) and plot the resultant energies versus temperature (*Mathematica* does this using the ListPlot command). Explain your results.

Problem 3

Modify the provided function to calculate the average magnetization (the sum of all spin values). Run the program at a variety of temperatures and discuss you results. **Note**: there are a variety of ways to do this in *Mathematica*...one suggestion is to look at the Flatten command an see how it could be useful.

Problem 4

To quantify the clustering of alignments with an Ising magnet, we can define a quantity called the **correlation function, c(r)**. To do this, take any two sites *i* and *j*, seperated by a distance *r*, and compute the product of thier states: if the product $s_i\, s_j$is positive, the dipoles are parallel, if it is negative, they are anti-parallel. If you average this quantity over *all* pairs within a certain fixed distance, you obtain a measure of the tendency of dipoles to be "correlated" over this distance. Effectively, you find the distance over which sites can "communicate" with other sites. To remove the effect of any overall magnetization of the system, you subtract off the square of the average *s*. Thus, the correlation function is given as

$$c(r) = \overline{s_i\, s_j} - \overline{s_i}^2,$$

where it is understood that the first term averages over all pairs at the fixed distance *r*.

a.) Modify the provided function to calculate the correlation function for the final state of the lattice, averaging over all pairs vertically and horizontally (not diagonally) seperated by *r* units of distance, where *r* ranges from 1 to half the width of the lattice.

b.) Run this program at a variety of temperatures above, at, and below the critical temperature (use a larger lattice near the critical temperature). Describe the behavior of the correlation function with respect to temperature.