Foreword

I used the c programming language to complete this task. I represented the system of spins as a `struct`, defined below in one dimension.

```
typedef struct ising_t
{
    float epsilon;
    float magnetic_field;
    float epsilon;
    int length;
    int *ensemble;
} ising_t;
```

Scaling this to two dimensions was more difficult because the `*ensemble` had to be transformed into `**ensemble`. It is possible to implement this in a single `struct` using `union ensemble {int *1d, int **2d};`, but in the code available on my `github` I have used two separate `structs` resulting in code duplication for many of the methods. I made this descision because I had not used `c` prior to this project and was unaware of the `union` keyword and its uses. I am describing this because you may notice discrepancies in the code snippets that I have include where `int *(*)ensemble = system -> ensemble;` is invoked to access the array of spins.

Question 1 a)

I selected three different temperatures, $1.0, 2.0$ and $3.0\epsilon/k$, and ran the metropolis algorithm for $1000N$ steps. At each of these temperatures the system was initialised in a random state and allowed to equilibrate.

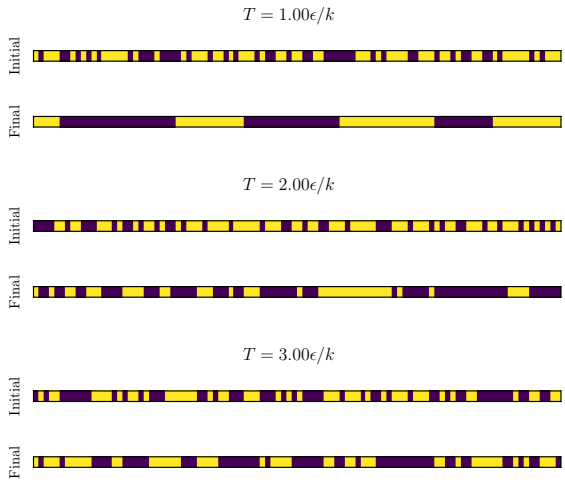

Figure 1: Each vertical pair of lines represents the spin state. The top one is the random initial state and the bottom one is the equilibrated final state.

We noticed that at lower temperatures the spin chunks were much larger than at higher temperatures. This is particularly pronounced between the $T = 3.0\epsilon/k$ and $T = 1.0\epsilon/k$ plots in figure 1. To evolve the system we used the version of the metropolis algorithm shown below,

```
/*
 * metropolis_step
 * -----------------------
 * Evolve the system by attempting to
 * flip a spin. The step is weighted
 * by the Boltzmann factor.
 *
 * parameters
 * ----------
 * ising_t *system: A struct
 *     enscapulating the information
 *     related to the system.
 */
void metropolis_step(ising_t *system)
{
    float temp = system -> temperature;
    int *ensemble = system -> ensemble;
    int length = system -> length;
    int spin = random_index(length);
    int change = 2*ensemble[spin]*(
        ensemble[modulo(spin+1,length)] +
        ensemble[modulo(spin-1,length)]);

    if ((change < 0)
        || (exp(-change/temp)>randn()))
    {
        ensemble[spin] *= -1;
    }
}
```

where, `rand` generates a random number in the range $[0, 1]$, and `modulo` is modified to produce positive input on negative numbers like the `python` implmentation. This is not the native `c` implementation. I used the `||` short circuit operator so that the second comparison was not evaluated on every call to the function.

Question 1 b)

I found that it was worth considering what the *basic unit* of the Ising model was. In the absence of an external magnetic field the energy is a function of the pairs. I start by considering the partition function of an individual pair. This is a two level system; either the pair are aligned or they are anti-aligned with the corresponding energies.

$$\begin{aligned} Z_i &= \sum_{s_i=\pm 1} \exp\left(-\frac{\varepsilon s_i s_{i+1}}{\tau}\right) \\ &= \exp\left(-\frac{\varepsilon}{\tau}\right) + \exp\left(\frac{\varepsilon}{\tau}\right) \\ &= 2\cosh\left(\frac{\varepsilon}{\tau}\right). \end{aligned} \tag{1}$$

Similarly to the para-magnetic case we can multiply the system partition functions of single constituents together

to get the partition function of the entire system. However, the condition to do this was that the constiutuents were independent, but the Ising model contains interactions. In the case of the Ising model the constituents that are independent are the pairs, not the individual spins. You may think think then that we only consider $N/2$ unique pairs but this is not the case. In a chain each spin is counted in two pairs so the power is still N.

A small detail that I skipped was what happens at the boundary. The two spins on the end of the chains are not (neccessarily) counted twice. In the limit of a very large chain of spins we can see that the boundary affect will not matter however, we got about this nuance in a much more interesting way by considering cyclic boundary conditions. That is to say that the spin on the far end of the chain is a neighbour to the spin at the start of the chain and vice versa.

Given the partition function $Z = (2\cosh(\varepsilon/\tau))^N$, we calculated the internal energy using,

$$\begin{aligned} U &= \tau^2 \partial_\tau \ln(Z) && (2)\\ &= \tau^2 \partial_\tau \ln\left(2\cosh\left(\frac{\varepsilon}{\tau}\right)^N\right) \\ &= N\tau^2 \partial_\tau \ln\left(2\cosh\left(\frac{\varepsilon}{\tau}\right)\right) \\ &= N\tau^2 \partial_\tau \left(2\cosh\left(\frac{\varepsilon}{\tau}\right)\right) \frac{1}{2\cosh\left(\frac{\varepsilon}{\tau}\right)} \\ &= N\tau^2 \partial_\tau \left(\frac{\varepsilon}{\tau}\right) \frac{\sinh\left(\frac{\varepsilon}{\tau}\right)}{\cosh\left(\frac{\varepsilon}{\tau}\right)} \\ &= -\varepsilon N \tanh\left(\frac{\varepsilon}{\tau}\right). && (3) \end{aligned}$$

We calculated the free energy of the system using,

$$\begin{aligned} F &= -\tau \ln Z && (4)\\ &= -\tau \ln\left(\left(2\cosh\left(\frac{\varepsilon}{\tau}\right)\right)^N\right) \\ &= -N\tau \ln\left(2\cosh\left(\frac{\varepsilon}{\tau}\right)\right) \\ &= -N\tau \ln\left(\exp\left(\frac{\varepsilon}{\tau}\right) + \exp\left(-\frac{\varepsilon}{\tau}\right)\right) \\ &= -N\tau \ln\left(\exp\left(\frac{\varepsilon}{\tau}\right)\left(1 + \exp\left(-2\frac{\varepsilon}{\tau}\right)\right)\right) \\ &= -N\tau \ln\left(\exp\left(\frac{\varepsilon}{\tau}\right)\right) - N\tau \ln\left(1 + \exp\left(-2\frac{\varepsilon}{\tau}\right)\right) \\ &= -N\varepsilon - N\tau \ln\left(1 + \exp\left(-2\frac{\varepsilon}{\tau}\right)\right). && (5) \end{aligned}$$

The entropy followed from the combination of Equation 3 and Equation 1 using Equation 5,

$$\begin{aligned} \tau\sigma &= F - U && (6)\\ &= -N\varepsilon \tanh\left(\frac{\varepsilon}{\tau}\right) + N\varepsilon + N\tau \ln\left(1 + \exp\left(-2\frac{\varepsilon}{\tau}\right)\right) \\ \sigma &= \frac{\varepsilon}{\tau}\left(1 - \tanh\left(\frac{\varepsilon}{\tau}\right)\right) + \ln\left(1 + \exp\left(-2\frac{\varepsilon}{\tau}\right)\right). && (7) \end{aligned}$$

Finally, we determined the specific heat using Equation 3 and Equation 5,

$$\begin{aligned} C &= \partial_\tau U && (8)\\ &= \partial_\tau \left(-N\varepsilon \tanh\left(\frac{\varepsilon}{\tau}\right)\right) \\ &= -N\varepsilon \partial_\tau \left(\frac{\varepsilon}{\tau}\right) \frac{1}{\cosh^2\left(\frac{\varepsilon}{\tau}\right)} \\ &= \frac{N\varepsilon^2}{\tau^2 \cosh^2\left(\frac{\varepsilon}{\tau}\right)}. && (9) \end{aligned}$$

Question 1 c)

We started by simulating a one dimensional Ising model with no external magnetic field, which we compared to the analytic expressions derived in the theory. As we described in theory we used periodic boundary conditions and chose to implement our models using a lattice size of one-hundred spins for the most part. We chose to use one-hundred spins because it evaluated fast on our device and was large enough to be interesting.

The Ising algorithm scales very non-linearly with the size of the lattice. This is because when taking a sample you want to run the system for enough iterations so that every spin has a chance to be visited many times and then average the result. In addition, operations such as finding the energy, require that you visit every spin and hence have linear time complexity independently.

Starting with our one dimensional model we equilibrated the system for multiple different temperatures and settled on using $1000N$ as the length of the loop. This was likely too many but we found that for low temperatures when the probability of a flip becomes small, a larger number of steps was required. For each temperature we started the chain of spins at a random temperature.

Once we had determined the numberic details of our simulations we proceeded to measure the physical properties of the system. To calculate the energy of the one dimensional case we employed the following algorithm:

Question 1 d)

Question 1 e)