0.1 Calculating canonical ensemble averages

```
import numpy as np
```

```
class Ensemble:
    def __init__(self, Es, lngs, name = 'Canonical ensemble', λ = None):
        self.Es = Es
        self.lngs = lngs
        self.name = name
        # Choose to scale the exponent of Z to be a convenient size.
        # This can be improved if needed by taking into account typical β*Es.
        self.λ = max(lngs) if λ is None else λ
    def Zλ(self, β):
        return np.sum(np.exp(-(np.outer(β, self.Es) - self.lngs + self.λ)), 1)
    def Z(self, β):
        return np.exp(self.λ) * self.Zλ(β)
    def p(self, β):
        return np.exp(-(β * self.Es - self.lngs + self.λ)) / self.Zλ(β)
    def average(self, f, β):
        return np.sum(f(self) * np.exp(-(np.outer(β, self.Es) - self.lngs + self.λ)), 1) / self.Zλ(β)
    def energy(self, β):
        return self.average(lambda ens: ens.Es, β)
    def energy2(self, β):
        return self.average(lambda ens: ens.Es**2, β)
    def heat_capacity(self, β):
        return self.energy2(β) - self.energy(β)**2
    def free_energy(self, β):
        return -np.log(self.Z(β)) / β
    def entropy(self, β):
        return β * self.energy(β) + np.log(self.Z(β))
```