The Problem

Assume that a mess in a college campus wants to open a cheese station similar to the egg counter. The cheese must be purchased at the counter of this station. The following contains the details of this scenario.

- Every weekday morning, the mess master must decide how much cheese to buy from the vendor.
- Each cheese packet has 100, 200, 300, 400, or 500 slices.
- Every workday, the mess committee can purchase only one packet of cheese or nothing at all.
- Each cheese slice is estimated to cost 10 rupees.
- Each cheese slice costs 12 rupees to purchase from the mess counter.
- During weekdays, the mess fridge can only hold 500 pieces of cheese slices.
- However, owing to the weekend cleaning routine, the fridge does not operate on weekends. Thus, there is no cheese counter on Saturdays and Sundays.
- Any cheese slices that haven't been sold by Friday evening will be wasted as they will not be consumable.
- If the number of slices purchased and the number of existing slices exceeds the capacity on a weekday, the extra inventory is wasted.
- The demand distribution on any given day is given as follows: For 100, 200, 300, 400, and 500 cheese slices, respectively, the corresponding probabilities are 0.15, 0.05, 0.3, 0.25, and 0.25.
- Because the mess committee cannot sell more cheese than the number of slices available, sales in a day are limited by the minimum demand and available slices.

Now giving the context, we need to maximize the total expected profit of the mess over the whole week. The environment is given by the Mess class. In this class, the demand on each day is given by demand value, and the corresponding probabilities are given by demand probs. On each weekday, the mess master can buy cheese from the following set $\{0, 100, 200, 300, 400, 500\}$ of cheese slices. So, the possible action spaces are given by the action space variable. The state is a tuple of the day of the week (or the weekend) and the starting cheese level for that day, for example ("Tuesday", 100). Next, get the next state reward is a method that calculates the next state and the reward along with the current state, the action, and the demand. The template also has another method called get transition prob that returns all possible transitions and rewards for a given state-action pair using the get next state reward method, together with the corresponding probabilities. Your task is to complete the functions of iterative policy evaluation and value iteration.

The example policy function is the demo policy. Assume all the policies are deterministic.

A policy is given in the form of a dictionary that maps states to action probabilities; for example, Policy = (‘Monday’, 0): 100: 0.4, 300: 0.6, (‘Tuesday’, 0): 100: 0.4, 300: 0.6. . . . And the value of states is given by a dictionary such as Values v = (‘Monday’, 0): 2884.0, (‘Tuesday’, 0): 2204.0, (‘Tuesday’, 100): 3204.0, ...

Just fill in the code for the missing functions in the template.