There are two options. If you find this stuff easy, I encourage you to try the second option (or try both). You will want to base the code off of `https://lectures.quantecon.org/jl/discrete_dp.html` and `https://github.com/jlperla/ECON407_2018/blob/master/notebooks/discrete_dp.ipynb`.

Option 1: Cake Eating

This is a variation on the `https://github.com/jlperla/ECON407_2018/blob/master/notebooks/discrete_dp.ipynb`.

A consumer is endowed with a cake with $s > 0$ slices. She

- Chooses $c \in \{0, 1, \dots s\}$ slices of cake to consume each period
- Gains utility $u(c) = c^\alpha$ from consuming those slides, where $\alpha > 0$. The consumer discounts future cake consumption with factor $\beta \in (0, 1)$.

At the end of a period, there is a probability $\rho \in [0, 1]$ that if any cake is left that one of the slices will expire. Hence, given that the cake starts at size s, and they consume $0 \leq c \leq s$ pieces of cake, the size of the cake next period is,

$$s' = \begin{cases} 0 & \text{if } s - c = 0 \text{ with probability } 1 \\ s - c & \text{if } s - c > 0 \text{ with probability } 1 - \rho \\ s - c - 1 & \text{if } s - c > 0 \text{ with probability } \rho \end{cases} \tag{1}$$

1. Map this problem into the DPP setup. In particular, you will need to define the R reward array and Q transition probability array.[1]
2. Write code to solve this using the `DiscreteDP` function. You should be able to parameterize this by the maximum size of cake.
3. Show 5 paths simulating the optimal consumption and cake sizes starting from the same initial cake size. For the parameter values feel free to play around with them to get interesting results. Perhaps you should start playing around with $s = 30$ as the initial cake $\alpha = 1/2$ as the coefficient on the utility function, $\rho = 0.1$, and $\beta = 0.9$
4. Plot it for $\alpha = 0.1$ and see how the optimal policy changes. Again, feel free to play around with the parameters to make it interesting.
5. Plot it for a much larger ρ and see how the optimal policy changes. Again, feel free to play around with the parameters to make it interesting.

An alternative formulation: Convert to the choice $a \equiv s - c$ instead of c. In that formulation,

- The payoff is $u(a) = (s - a)^\alpha$
- Feasibility is $0 \leq a \leq s$

$$s' = \begin{cases} 0 & \text{if } a = 0 \text{ with probability } 1 \\ a & \text{if } a > 0 \text{ with probability } 1 - \rho \\ a - 1 & \text{if } a > 0 \text{ with probability } \rho \end{cases} \tag{2}$$

[1] Recall: you can set $R(s, c) = -\infty$ if an action is not feasible (i.e. $c > s$).

Option 2: McCall Search as a DPP Problem

In the code and algebra of `https://lectures.quantecon.org/jl/mccall_model.html`, we solved the model with value function iteration. However, the search model could also fit into the dynamic programming problem described in `https://lectures.quantecon.org/jl/discrete_dp.html` and implemented with modified code in `https://github.com/jlperla/ECON407_2018/blob/master/notebooks/discrete_dp.ipynb`.

In this question, I want you to define the McCall model in the notation of the DPP notes (i.e. a specification of the `Q` and `R`, the state space, etc), and then solve it using the DPP library. While you have flexibility on how you accomplish this, you will need to rewrite the problem bit to better match the general approach. One suggestion (of many),

- As before, let the possible wages be $w_1, \dots w_n$.
- Instead of a separate $i = 1, \dots n$ vector for the wage state and U for the unemployment state, put them together and standardize on $n + 1$ as the "unemployment" state. Therefore, if $s = i$ for $i = 1, \dots, n$ they get w_i, and if $s = n + 1$ they get c income.
- The decision problem is then: given a state $s \in 1, \dots n + 1$, choose to accept or reject the job. If they accept the job they get immediate wage w_i and move onto the next period (potentially losing their job with probability α) and if they reject the job they get c and have γ chance of getting a job offer next period. For simplicity, we can denote $w_{n+1} = c$, then the Bellman equation is

$$V(i) = \max \{u(w_{n+1}) + \beta \left(\gamma \mathbb{E}\left[V(i')\right] + (1-\gamma)V(n+1)\right) \tag{3}$$

$$u(w_i) + \beta \left((1-\alpha)V(i) + \alpha V(n+1)\right)\} \tag{4}$$

- Where $\mathbb{E}\left[V(i')\right] = \sum_{i'=1}^{n} V(i')p(w_{i'})$ given the probabilities of the wage offers as specified in the original code.
- Another way to think about this specification: by accepting/rejecting the worker is able to choose to move to state $n + 1$ with certainty next period (and, hence, she changes the transition probabilities in `Q` conditional on her action).

Given your specification of the McCall Search model in the DPP form, try to implement similar figures to those in `https://github.com/jlperla/ECON407_2018/blob/master/notebooks/discrete_dp.ipynb`.