2 -(a)

Expected Value at each iteration could be calculated as

Max(Reward + Discount\_factor \* (p(stay) \* Reward+ p(move)\*Next\_Reward , -250 + Discount\_factor \* New

The method is keep update the expected values for each states until the difference between updated one is less than 0.00001.

The result is

New = 800.5315300923165

Use1 = 778.3683759820096

Use2 = 643.222218799097

Use3 = 556.1234953283804

Use4 = 502.8359296100837

Use5 = 475.84593119889684

Use6 = 470.47837708308487

Use7 = 470.47837708308487

Use8 = 470.47837708308487

Dead = 470.47837708308487

2 - (b)

The replace value is -250 + 0.9\*n = 470.47837708308487

The utility value for state Use8 is 443.43053937477646.

The utility value for state Use7 is 453.43053937477634

The utility value for state Use6 is 463.4305393747764

The utility value for state Use5 is 475.8459387268918

The New and Use1 ~4 has bigger utility value than Use5.

Use6,7,8 should take action "Replace".

Use1~5 should take action "Use".

2 - (c)

The method is that computing the expected values in case we replace to New, and in case we replace to Use1 or Use2 at the same time. Let's say p is the cost of change to Use1 or Use2. Firstly, we will assign the p value as the mean of the minimum possible value and maximum possible value. If the expected value of replacing to use one is bigger, we will change the minimum possible value to this p, and then choose new p as the mean of minimum and maximum possible value.

If smaller, then change the maximum value to this p, and then get new p. Until the expected value of replacing to used one is bigger, and the difference is smaller than 0.001, we will iterate to find the best P.

The calculation for expected value is same as above, but the replacement part would be changed to

-p + 0.9\*(0.5\*Use1 + 0.5\*Use2)

When the cost for replacing to Use1 or Use2 is 169, people will choose it.

2-(d)

I choose the several discount factors, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99, 0.999, 0.9999. If the factor is less than 0.9, they have different expected values in each state. When the factor is 0.9, Use6,7,8 and Dead state have same expected value. When the factor is 0.99, Use4,5,6,7,8 and Dead state have same expected value. When the factor is 0.999 and 0.9999, also Use4,5,6,7,8 and Dead state have same expected value.

Thus, there is a point that the optimal policy is not changed.

The method to find the best beta (sufficiently large) is pretty same as the one I used in above question. I found that from 0.9 to 0.99, there is a point that the optimal policy is set. So, I set the minimum value as 0.9, and maximum as 0.99. Choose the first beta as the mean of minimum and maximum. We detect that the optimal policy would be Use 4,5,6,7,8 and dead have same expected value. So, if the expected value of Use4 and Use5 are different, we will change the minimum value to this beta value, and get the new beta. If same, we will change the maximum value to this beta value, and update to new beta.

The best beta is 0.9585.