RL Assignment 2

Suryatej Reddy
2016102

Question 2:

In question 2, i used the bellman equations given in the equation to create a matrix of the form Ax = b. Then used a linear equation solver in python to generate the state value functions.

```
array([[ 3.3,  8.8,  4.4,  5.3,  1.5],
       [ 1.5,  3. ,  2.3,  1.9,  0.5],
       [ 0.1,  0.7,  0.7,  0.4, -0.4],
       [-1. , -0.4, -0.4, -0.6, -1.2],
       [-1.9, -1.3, -1.2, -1.4, -2. ]])
```

Question 4:

In question 4, the optimal bellman equations are non linear. So we have to consider a different equation for each of the action states and find the maximum over them. The best way to do this to create an inequality of the type AX>=b and solve the inequality. We know this is current because the optimal state value function will be greater than or equal to the q at each action. Doing this we get the optimal state value function as :

```
array([[22. , 24.4, 22. , 19.4, 17.5],
       [19.8, 22. , 19.8, 17.8, 16. ],
       [17.8, 19.8, 17.8, 16. , 14.4],
       [16. , 17.8, 16. , 14.4, 13. ],
       [14.4, 16. , 14.4, 13. , 11.7]])
```

We can use this matrix to generate the optimal policy. Doing that, we get the optimal policy as:

```
[['Right ', 'Left Right Up Down ', 'Left ', 'Left Right Up Down ', 'Left '],
 ['Right Up ', 'Up ', 'Left Up ', 'Left ', 'Left '],
 ['Right Up ', 'Up ', 'Left Up ', 'Left Up ', 'Left Up '],
 ['Right Up ', 'Up ', 'Left Up ', 'Left Up ', 'Left Up '],
 ['Right Up ', 'Up ', 'Left Up ', 'Left Up ', 'Left Up ']]
```

Question 6:

POLICY ITERATION

To solve the bug mentioned in the questions, i took a stochastic setting and created and array to store p(a|s) for each of the 4 actions. After following the algorithm mentioned in the textbook, we can observe some logs as given below:

```
Updated Value of State 1 From -13.999312424461952 To -1.0
Updated Value of State 2 From -19.999011518162757 To -2.0
Updated Value of State 3 From -21.99891199249635 To -3.0
Updated Value of State 4 From -13.999312424461952 To -1.0
Updated Value of State 5 From -17.999156254598965 To -2.0
Updated Value of State 6 From -19.99908388638086 To -3.0
Updated Value of State 7 From -19.999094361586472 To -14.999422844683945
Updated Value of State 8 From -19.999011518162753 To -2.0
Updated Value of State 9 From -19.99908388638086 To -3.0
Updated Value of State 10 From -17.99922696784339 To -14.999422844683943
Updated Value of State 11 From -13.999422844683945 To -1.0
Updated Value of State 12 From -21.998911992496346 To -3.0
Updated Value of State 13 From -19.99909436158647 To -14.999422844683943
```

As we can see, in most of these states the state value function is updated for the better. The rest of them are updated in subsequent iterations. Finally after convergence, we get

```
array([[ 0., -1., -2., -3.],
       [-1., -2., -3., -2.],
       [-2., -3., -2., -1.],
       [-3., -2., -1.,  0.]])
```

```
[['-', 'Left ', 'Left ', 'Left Down '],
 ['Up ', 'Left Up ', 'Left Right Up Down ', 'Down '],
 ['Up ', 'Left Right Up Down ', 'Right Down ', 'Down '],
 ['Right Up ', 'Right ', 'Right ', '-']]
```

VALUE ITERATION

Following the algorithm mentioned in the textbook, we get the following logs:

```
Updated Value of State 1 From -1 To -1
Updated Value of State 2 From -1 To -2
Updated Value of State 3 From -1 To -2
Updated Value of State 4 From -1 To -1
```

```
Updated Value of State 5 From -1 To -2
Updated Value of State 6 From -1 To -2
Updated Value of State 7 From -1 To -2
Updated Value of State 8 From -1 To -2
Updated Value of State 9 From -1 To -2
Updated Value of State 10 From -1 To -2
Updated Value of State 11 From -1 To -1
Updated Value of State 12 From -1 To -2
Updated Value of State 13 From -1 To -2
Updated Value of State 14 From -1 To -1
```

And the following results:

```
array([[ 0, -1, -2, -3],
       [-1, -2, -3, -2],
       [-2, -3, -2, -1],
       [-3, -2, -1,  0]])
```

```
[['-', 'Left ', 'Left ', 'Left Down '],
 ['Up ', 'Left Up ', 'Left Right Up Down ', 'Down '],
 ['Up ', 'Left Right Up Down ', 'Right Down ', 'Down '],
 ['Right Up ', 'Right ', 'Right ', '-']]
```