1a->

```
-2.700 0.000 -0.573 27.000
-2.975 -2.981 -2.779 -0.573
-2.997 -2.997 0.000 -2.802
-2.997 -2.997 -5.400 -2.995
- - e -
u e e u
w w - u
w w - e
Steps taken -
3
```

Since discount factor is very less, states that are far away from the start state have very little contribution in calculating the utility. So, it goes to the state without thinking far ahead, and goes to the nearest positive end state.

1b->

```
-2.700 0.000 22.895 27.000
10.114 15.536 19.645 22.895
7.872 11.521 0.000 19.243
4.599 6.337 -5.400 13.325
- - e -
e e e u
e u - u
u u - u
Steps taken -
16
```

Since discount factor is high here, it looks into the future properly and goes where the expected reward is highest given the step cost.

2a->

```
-2.700 0.000 1373.322 27.000
1373.322 1373.322 1373.322 1373.292
1373.322 1373.322 0.000 1373.290
1373.322 1373.322 -5.400 1373.258
- - w -
d e w d
e u - u
```

```
u w - e
Steps taken -
70
```
Since here step cost value is very high and positive, the
environment is very suitable to agent and it does not
want to come out of environment, therefore it tries to
go out of matrix, and stays at its position with 0.8
probability while gaining reward of 1373.322. There are many
other optimal policies for this case. For example
instead of down, it could try going to left.

2b->

```
-2.700 0.000 19.164 27.000
-2.452 5.127 12.958 19.164
-8.438 -2.427 0.000 12.192
-14.488 -9.318 -5.400 4.130
- - e -
e e e u
u u - u
u u - u
Steps taken -
13
```
Its almost same as the policy obtained with step cost =
-2.7. The difference is at (2, 0), it went upwards while
when step cost was -2.7, it went rightwards. Reason
may be because the step cost is higher here, so it takes the risk of ending up in the end
state with less positive reward. If it would have gone to right, then
there would be higher probability to go down to (3, 1)
from which it would have tried to go up again, thereby
Increasing number of steps and higher unit step cost.

2c->

```
-2.700 0.000 17.298 27.000
-8.667 -0.065 9.615 17.298
-16.137 -9.323 0.000 8.666
-20.924 -13.263 -5.400 -0.467
- - e -

e e e u
u u - u
```

```
e e - u
Steps taken -
16
```

Since the step cost is more negative here, the
expected cost to reach to most rewarding state (0, 0)
is more higher is it cancels out the reward agent would
get there. It was better for the agent to go to (0, 0) in
less steps and get less reward as the less cost is to
reach there and reward-cost value would be higher.

2d->

```
-2.700 0.000 -10.668 27.000
-40.363 -73.555 -40.480 -10.668
-73.555 -75.608 0.000 -44.146
-75.608 -42.960 -5.400 -39.548
- - e -
u w e u
u d - u
e e - w
Steps taken -
7
```

As the step cost is very high here, the environment is
very painful to the agent and it wants to get out of it
as soon as possible, so it goes to the nearest end state
possible even if it has negative reward.