The following was the output when the Discount factor was 0.1 a)

0	6.5	0	-1.381	65.0	 0	
1	-6.112	-7.132	-6.692	-1.381	 0	
2	-7.132	-7.214	0 -6.7	46 0		
3	-7.215	-7.221	-13.0	-7.217	0	
4	0	0	0	0		0
> -	-					

Since the discount factor is less, so the reward keeps diminishing with time and hence, the program converges to a negative value, taking only 6 iterations.

b) The following is the output of the code when discount factor was 0.99

0	6.5	0	55.119	65.0	0	
1	25.668	37.409	47.293	55.119	0	
2	19.158	27.774	0 46.3	27 0		
3	11.311	15.33	-13.0	32.08	0	
4	0	0	0	0		0

Here, the discount factor is very high, so the program undergoes a lot of iterations and converges finally to a high positive value.

^{^ &}lt; > ^

^{^ ^} _ ^

^{^ &}lt; - >

> ^ _ ^

^{^ ^} _ ^

2. This part b

a)

0	6.5	0	6499.99	65.0	0	
1	6499.99	6499.99	6499.99	6499.99		0
2	6499.99	6499.99	0	6499.99		0
3	6499.99	6499.99	-13.0	6499.99		0
4	0	0	0	0		0

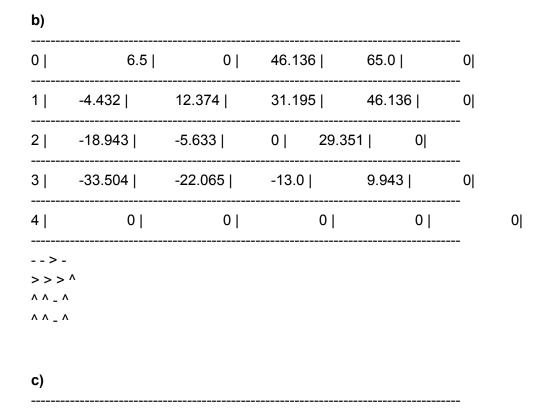
--<-

VVV

VV-V

V < - >

Since the step cost is very high, the bot gets a great reward at each step and hence tries to take a long route to try and accumulate the highest possible reward value. This results in a very massive final utility value.



0	6.5	0				
1	-12.328		23.147		0	
2	-31.236		0			
	-48.99					
	0					0
^ ^ _ / ^ > _ /						
	6.5					
1	-12.328					
2	-31.236		0			
3	-48.99	-31.827	-13.0	-1.125	0	
4	0	0	0	0		0

^ > - ^

For parts, (b), (c) and (d) the step cost keeps decreasing. This results in lesser iterations and also a lesser utility value on convergence.