MBAN
Decision Analysis Under Uncertainty

HW3

Team Members:

Pranav Mehta - 19184282

Zhuji Zhang - 65457343

Duo Xu - 91043547

Q1 (a): Machine Maintenance Problem Formulation

1. States (S)

- Running
- Broken

2. Actions (A)

- If the machine is Running:
    - Maintain
    - Do Nothing
- If the machine is Broken:
    - Repair
    - Replace

3. Transition Probabilities (P(j|s, a)):

- From Running:
    - Maintain:
    - P(Running | Running, Maintain) = 0.6
    - P(Broken | Running, Maintain) = 0.4
    - Do Nothing:
    - P(Running | Running, Do Nothing) = 0.35
    - P(Broken | Running, Do Nothing) = 0.65

- From Broken:

    - Repair:

    - P(Running | Broken, Repair) = 0.6

    - P(Broken | Broken, Repair) = 0.4

    - Replace:

    - P(Running | Broken, Replace) = 1.0

    - P(Broken | Broken, Replace) = 0.0

4. Rewards (R(s, a, j)):

- From Running:

    - Maintain, Running: +70 (Profit 100 - Maintenance Cost 30)

    - Maintain, Broken: -30 (Maintenance Cost 30)

    - Expected Rewards for Running, Maintain = 70*0.6 + (-30)*0.4 = 30

    - Do Nothing, Running: +100 (Profit 100)

    - Do Nothing, Broken: 0 (No Profit)

    - Expected Rewards for Running, Do Nothing = 0*0.65 + 100*0.35 = 35

- From Broken:

    - Repair, Running: 40 (Profit 100 – Repair Cost 60)

    - Repair, Broken: -60 (Repair Cost 60)

    - Expected Rewards for Broken, Repair = 40*0.6 + (-60)*0.4 = 0

    - Replace, Running: -10 (Profit 100 – Replacement Cost 110)

    - Replace, Broken: -110 (Replacement Cost 110)

    - Expected Rewards for Broken, Replacement = -110*0 + (-10)*1 = -20

5. Summarized Table for Expected Rewards Calculation:

s	a	j	P(j|s,a)	r(s,a,j)	Expected Rewards
Broken	repair	Broken	0.4	-60	0
Broken	repair	Running	0.6	40	
Broken	replace	Broken	0	-110	-10
Broken	replace	Running	1	-10	
Running	maintain	Broken	0.4	-30	30
Running	maintain	Running	0.6	70	
Running	wait	Broken	0.65	0	35
Running	wait	Running	0.35	100	

5. Bellman Equations:

Let V_t(s) be the value function for state s at time t. The Bellman equations are:

- Before the Final Period (t < N):

- When Running:

Vt(Running) = max {

Maintain: 30 + 0.6 * Vt+1(Running) + 0.4 * Vt+1(Broken),

Wait: 35 + 0.35 * Vt+1(Running) + 0.65 * Vt+1(Broken) }

Vt(Running) represent the value when the machine is running at the stage t

- Under the action **Maintain**, there is a **60%** probability that the machine continues running until the start of the next period, and a **40%** probability it becomes broken. The expected reward of the current period is **+30**.
- Under the action **Wait**, there is a 35% probability of staying Running and a 65% probability of becoming Broken. The expected reward of the current period is +35.
- We choose the action that maximizes the value for Vt(Running)

- When Broken:

Vt(Broken) = max {

Repair: 0 + 0.6 * Vt+1(Running) + 0.4 * Vt+1(Broken),

Replace: -10 + Vt+1(Running) }

Vt(Broken) represents the value when the machine is broken at stage *t*

- Under the action **Repair**, there is a **60%** probability that the machine becomes Running and a **40%** probability it remains Broken. The expected reward of the current period is **0**.
- Under the action **Replace**, the machine is guaranteed to transition to Running at the next stage. The expected reward of the current period is **-10**.
- We choose the action that gives the highest value for Vt(Broken)

- At the Final Period (t = N):

VN(Running) = Max {30,35} = 35

VN(Broken) = Max {0,-10} = 0

- in the final week, the only consideration is the immediate reward. Between Maintain (+30) and **Wait (+35)**, Wait yields the maximum reward of **+35**
- If the machine is Broken in the final week, Repair provides a reward of 0 and Replace incurs a loss of -10. Therefore, **Repair** is the better action, resulting in **0**.

Q1 (b): Solving Machine Maintenance Problem with R

N=10:

```
            Week 1      Week 2      Week 3      Week 4      Week 5      Week 6
Running Maintenance Maintenance Maintenance Maintenance Maintenance Maintenance
Broken      Replace     Replace     Replace     Replace     Replace     Replace
            Week 7      Week 8      Week 9       Week 10
Running Maintenance Maintenance Maintenance No Maintenance
Broken      Replace     Replace     Replace         Repair
```

```
Time Step: 1
    State  Value
1 Running 200.31
2  Broken 171.74

Time Step: 2
    State  Value
1 Running 181.74
2  Broken 153.16

Time Step: 3
    State  Value
1 Running 163.16
2  Broken 134.60

Time Step: 4
    State Value
1 Running 144.6
2  Broken 116.0

Time Step: 5
    State Value
1 Running 126.0
2  Broken  97.5
```

```
Time Step: 6
    State  Value
1 Running 107.50
2  Broken  78.76

Time Step: 7
    State Value
1 Running 88.76
2  Broken 60.60

Time Step: 8
    State Value
1 Running  70.6
2  Broken  41.0

Time Step: 9
    State Value
1 Running    51
2  Broken    25

Time Step: 10
    State Value
1 Running    35
2  Broken     0
```

Optimal Policy for N=10:

For the Running state, "Maintain" is optimal for most weeks, followed by "Do Nothing" in the final week. For the Broken state, "Replace" is the preferred action, with "Repair" only being chosen in the final week.

Throughout the timeline, the values for both "Running" and "Broken" states show a declining trend, reflecting the diminishing opportunities for future rewards as time progresses. The "Running" state starts with a higher value (200.31 in week 1) due to its ability to generate positive rewards, while the "Broken" state starts lower (171.74) because of penalties and lack of immediate profits. Over time, the values for both states decrease. This steep decline indicates that as the timeline approaches the end, the chances to recover maintenance or repair costs diminish, reducing the expected future rewards accordingly.

N=20:

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
Running	Maintenance	Maintenance	Maintenance	Maintenance	Maintenance	Maintenance
Broken	Replace	Replace	Replace	Replace	Replace	Replace
	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Running	Maintenance	Maintenance	Maintenance	Maintenance	Maintenance	Maintenance
Broken	Replace	Replace	Replace	Replace	Replace	Replace
	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18
Running	Maintenance	Maintenance	Maintenance	Maintenance	Maintenance	Maintenance
Broken	Replace	Replace	Replace	Replace	Replace	Replace
	Week 19	Week 20				
Running	Maintenance	No Maintenance				
Broken	Replace	Repair				

Time Step: 1		Time Step: 6		Time Step: 11		Time Step: 16	
	State	Value	State	Value	State	Value	State	Value
1	Running	386.02	Running	293.16	Running	200.31	Running	107.50
2	Broken	357.45	Broken	264.59	Broken	171.74	Broken	78.76
Time Step: 2		**Time Step: 7**		**Time Step: 12**		**Time Step: 17**	
	State	Value	State	Value	State	Value	State	Value
1	Running	367.45	Running	274.59	Running	181.74	Running	88.76
2	Broken	338.88	Broken	246.02	Broken	153.16	Broken	60.60
Time Step: 3		**Time Step: 8**		**Time Step: 13**		**Time Step: 18**	
	State	Value	State	Value	State	Value	State	Value
1	Running	348.88	Running	256.02	Running	163.16	Running	70.6
2	Broken	320.31	Broken	227.45	Broken	134.60	Broken	41.0
Time Step: 4		**Time Step: 9**		**Time Step: 14**		**Time Step: 19**	
	State	Value	State	Value	State	Value	State	Value
1	Running	330.31	Running	237.45	Running	144.6	Running	51
2	Broken	301.73	Broken	208.88	Broken	116.0	Broken	25
Time Step: 5		**Time Step: 10**		**Time Step: 15**		**Time Step: 20**	
	State	Value	State	Value	State	Value	State	Value
1	Running	311.73	Running	218.88	Running	126.0	Running	35
2	Broken	283.16	Broken	190.31	Broken	97.5	Broken	0

Optimal Policy for N=20:

The Policy when N=20 is Similar to when N=10: For the Running state, "Maintain" is optimal for most weeks, followed by "Do Nothing" in the final week. For the Broken state, "Replace" is the preferred action, with "Repair" only being chosen in the final week.

For N=20, the values for both "Running" and "Broken" states are initially higher compared to N=10 due to the extended time available to accumulate rewards, but similarly declines as N=10 overtime.

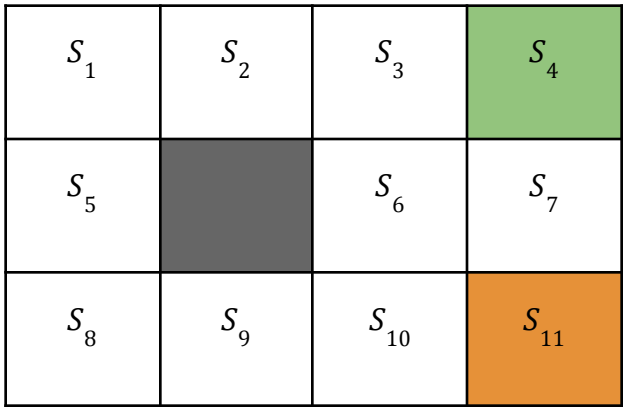
Conclusion for Q1-b

In conclusion, the optimal policy is to Maintain when Running and Replace when Broken. However, in the final period, the policy shifts to Wait when Running and Repair when Broken, as there are no more future rewards to consider.

Q2 (a): Robot Navigation Problem Formulation

States: $\{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}\}$

S_4 and S_{11} are absorbing states, the robot will stay there once it reaches these two states and the game will be over.



Actions: {Up, Down, Left, Right}
A_1: {D, R}, A_2: {L, R}, A_3: {D, L, R}, A_5: {U, D}, A_6: {U, D, R}, A_7: {U, D, L}, A_8: {U, R}, A_9 : {L, R}, A_{10}: {U, L, R}, $A_4 = A_{11}$ = No actions

Rewards:

	U	D	L	R
1,1	-9999	-1	-9999	-1
1,2	-9999	-9999	-1	-1
1,3	-9999	-1	-1	-1
1,4	1	1	1	1
2,1	-1	-1	-9999	-9999
2,2	-9999	-9999	-9999	-9999
2,3	-1	-1	-9999	-1
2,4	-1	-1	-1	-9999
3,1	-1	-9999	-9999	-1
3,2	-9999	-9999	-1	-1
3,3	-1	-9999	-1	-1
3,4	-1	-1	-1	-1

1. Terminal Rewards: +1 and −1

- The terminal states (green and orange cells) have been assigned normalized rewards of +1 and −1, respectively, instead of the original +100 and −100. This choice is consistent with Hint 2, which states that this problem ends when the robot reaches a terminal state (absorbing state), unlike the infinite-horizon MDP example where the robot continues to collect rewards indefinitely.

Green Terminal Cell (1,4):

- The reward of +1 reflects successful task completion and aligns with the absorbing state structure.
- Using +1 instead of +100 avoids large state values dominating the value function and maintains numerical stability in calculations.

Orange Terminal Cell (3,4):

- The reward of −1 represents failure and discourages the robot from reaching this state.
- Normalizing to −1 (instead of −100) ensures consistent scaling with movement costs and simplifies interpretation of the value function.

2. Movement Costs: −1

- Every valid action in non-terminal, non-wall cells is assigned a movement cost of -1. This reflects the cost incurred for each step the robot takes, as outlined in the problem statement.
- The movement cost incentivizes the robot to minimize unnecessary steps and prioritize efficient paths to the terminal states.

3. Penalties for Invalid Actions: −9999

- Actions that attempt to move into a wall, border, or inaccessible cell are assigned a large negative reward of −9999-9999. This penalty strongly discourages the robot from selecting invalid actions and is consistent with **Hint 1**, which suggests excluding such actions from the feasible action set by assigning them a large negative reward.

4. Wall Cell: (2,2)

- The wall cell is inaccessible, and all actions attempting to move into or through it are penalized with a reward of −9999-9999, ensuring the robot treats it as a completely blocked region.
- This aligns with the problem's structure, where walls act as barriers that restrict movement**.**

Bellman Equation:

$$V(s) = a\in\{U, D, L, R\}max[r(s, a) + \delta j\in S\sum p(j|s, a)V(j)], \forall s\in S$$

$$= a\in\{U, D, L, R\}max[r(s, a) + 0.99j\in S\sum p(j|s, a)V(j)], \forall s\in S$$

Transition Probabilities for action up:

p(j, |s, a) = p, if there are two valid perpendicular directions

= 1-2p, if there is only one or no valid perpendicular direction

=if perpendicular directions are invalid the robot will stay in the same cell with the corresponding probability

$(1,1)$	$(1,2)$	$(1,3)$	$(1,4)$
$(2,1)$	$(2,2)$	$(2,3)$	$(2,4)$
$(3,1)$	$(3,2)$	$(3,3)$	$(3,4)$

The above image should be used as a reference for navigating the table below, which provides the transition probabilities for the action 'Up'.

Action Up					
(Row,Col)	Up	Left	Right	Total Stay Prob	Prob to move
(1,1)	1-2p (Stay)	p (Stay)	p	1-p	p (right)
(1,2)	1-2p (Stay)	p	p	1-2p	p (left),p(right)
(1,3)	1-2p (Stay)	p	p	1-2p	p (left),p(right)
(1,4)	0	0	0	1	0
(2,1)	1-2p	p (Stay)	p (Stay)	2p	1-2p (up)
(2,2)	NA	NA	NA	NA	NA
(2,3)	1-2p	p (Stay)	p	p	1-2p (up), p(right)
(2,4)	1-2p	p	p (Stay)	p	1-2p (up), p(left)
(3,1)	1-2p	p (Stay)	p	p	1-2p (up), p(right)
(3,2)	1-2p (Stay)	p	p	1-2p	p (left),p(right)
(3,3)	1-2p	p	p	0	1-2p (up), p(left),p(right)
(3,4)	0	0	0	1	0

Action: UP													
FROM\TO	1,1	1,2	1,3	1,4	2,1	2,2	2,3	2,4	3,1	3,2	3,3	3,4	sum
1,1	0.98	0.02	0	0	0	0	0	0	0	0	0	0	1
1,2	0.02	0.96	0.02	0	0	0	0	0	0	0	0	0	1
1,3	0	0.02	0.96	0.02	0	0	0	0	0	0	0	0	1
1,4	0	0	0	1	0	0	0	0	0	0	0	0	1
2,1	0.96	0	0	0	0.04	0	0	0	0	0	0	0	1
2,2	0	0	0	0	0	1	0	0	0	0	0	0	1
2,3	0	0	0.96	0	0	0	0.02	0.02	0	0	0	0	1
2,4	0	0	0	0.96	0	0	0.02	0.02	0	0	0	0	1
3,1	0	0	0	0	0.96	0	0	0	0.02	0.02	0	0	1
3,2	0	0	0	0	0	0	0	0	0.02	0.96	0.02	0	1
3,3	0	0	0	0	0	0	0.96	0	0	0.02	0	0.02	1
3,4	0	0	0	0	0	0	0	0	0	0	0	1	1

The above table is an extension of the previous table. Here, we have used the value p=0.02p = 0.02.

Q2 (b) ***R Code to Find the Optimal Solution where p=0.02***

	State	Value.to.Go	Optimal.Policy
1	(1,1)	93.74	Right
2	(1,2)	95.82	Right
3	(1,3)	97.88	Right
4	(1,4)	100.00	Terminal
5	(2,1)	91.72	Up
6	(2,2)	-999999.00	Wall
7	(2,3)	95.86	Up
8	(2,4)	97.88	Up
9	(3,1)	89.68	Up
10	(3,2)	87.89	Right
11	(3,3)	89.86	Up
12	(3,4)	-100.00	Terminal

Optimal Policy for p = 0.02

Right	Right	Right	Terminal
Up	Wall	Up	Up
Up	Right	Up	Terminal

Value Function for p = 0.02

93.74	95.82	97.88	100
91.72	-999999	95.86	97.88
89.68	87.89	89.86	-100

Optimal Policy: Mostly **Right** and **Up**.

Value to Go: Higher due to low inaccuracy, allowing the robot to follow a near-optimal path.

Q2 (c) ***R Code to Find the Optimal Solution where p=0.1***

	State	Value.to.Go	Optimal.Policy
1	(1,1)	92.05	Right
2	(1,2)	94.78	Right
3	(1,3)	97.24	Right
4	(1,4)	100.00	Terminal
5	(2,1)	89.66	Up
6	(2,2)	-999999.00	Wall
7	(2,3)	95.05	Up
8	(2,4)	97.24	Up
9	(3,1)	87.01	Up
10	(3,2)	84.67	Left
11	(3,3)	83.76	Left
12	(3,4)	-100.00	Terminal

Optimal Policy for p = 0.1

Right	Right	Right	Terminal
Up	Wall	Up	Up
Up	Left	Left	Terminal

Value Function for p = 0.1

92.05	94.78	97.24	100
89.66	-999999	95.05	97.24
87.01	84.67	83.76	-100

Optimal Policy: Mostly ***Right*** and ***Up***.

Policy Change: The robot switches to *Left* in *states 9* and *10* to avoid the orange cell.

Value-to-Go: Lower due to higher inaccuracy, leading to a riskier and less optimal outcome.

Analysis:

1. Optimal Policy

- **For p=0.02:**
  - The robot mostly takes the shortest path to the green terminal cell (+100 reward), prioritizing "Right" and "Up" movements.
  - In the bottom row, the robot uses the "Up" action more frequently, likely due to lower inaccuracy.
- **For p=0.1:**
  - The robot's actions become more conservative, especially near the orange terminal cell (-100 penalty). It avoids risky movements to minimize the likelihood of reaching the penalty cell.
  - In the bottom row, "Left" is now chosen instead of "Up," indicating a shift to safer actions.

2. ***Value Function***

- **For p=0.02:**
  - Values across states are higher, reflecting the robot's ability to navigate more efficiently and avoid penalty states.
  - States near the green terminal cell have values close to 100, as the robot reliably reaches the reward state.
  - The state near the orange terminal cell has a relatively high value despite proximity, indicating confidence in avoiding the penalty.
- **For p=0.1:**
  - Values decrease across most states, indicating reduced confidence in reaching the green terminal and increased risk of entering penalty states.
  - The bottom row experiences significant drops in value due to the higher probability of deviating into the penalty state.

Summary:

1. **Policy Shift**: The robot shifts from aggressive, shortest-path strategies to safer, more conservative actions.
2. **Value Reduction**: Expected rewards decrease with higher inaccuracy due to increased risks of penalties and inefficiency.
3. **Risk Management**: The robot's optimal policy becomes more about avoiding penalties than maximizing immediate rewards.