10.

Aim:

The aim of this program is to implement the Value Iteration algorithm for a given MDP using Python programming language.

Algorithm:

Initialize the value function for each state to zero.

Repeat until convergence or maximum iterations:

For each state, calculate the expected value of each possible action, using the current value function.

Update the value function for each state to the maximum expected value across all possible actions.

Calculate the maximum difference between the new and old value functions.

If the maximum difference is less than a predefined threshold, stop the iteration.

Return the optimal value function and policy.

program

```
actions = (0, 1)

states = (0, 1, 2, 3, 4)

rewards = [-1, -1, 10, -1, -1]

gamma = 0.9

probs = [

[[0.9, 0.1], [0.1, 0.9], [0, 0], [0, 0], [0, 0]],

[[0.9, 0.1], [0, 0], [0.1, 0.9], [0, 0], [0, 0]],

[[0, 0], [0, 0], [0, 0], [0, 0], [0, 0]],

[[0, 0], [0, 0], [0.9, 0.1], [0, 0], [0.1, 0.9]],

[[0, 0], [0, 0], [0, 0], [0.9, 0.1], [0.1, 0.9]],

]

max_iter = 10000

delta = 1e-400

V = [0, 0, 0, 0, 0]

pi = [None, None, None, None, None]

for i in range(max_iter):

max_diff = 0

V_new = [0, 0, 0, 0, 0]

for s in states:
```

```
max_val = 0
for a in actions:
val = rewards[s]
for s_next in states:
val += probs[s][s_next][a] * (gamma * V[s_next])
max_val = max(max_val, val)
if V[s] < val:
pi[s] = actions[a]
V_new[s] = max_val
max_diff = max(max_diff, abs(V[s] - V_new[s]))
V = V_new
if max_diff < delta:
break
```

Output:

Optimal value function: [3.308, 7.746, 11.000, 7.746, 3.308]

Optimal policy: [1, 0, None, 1, 1]