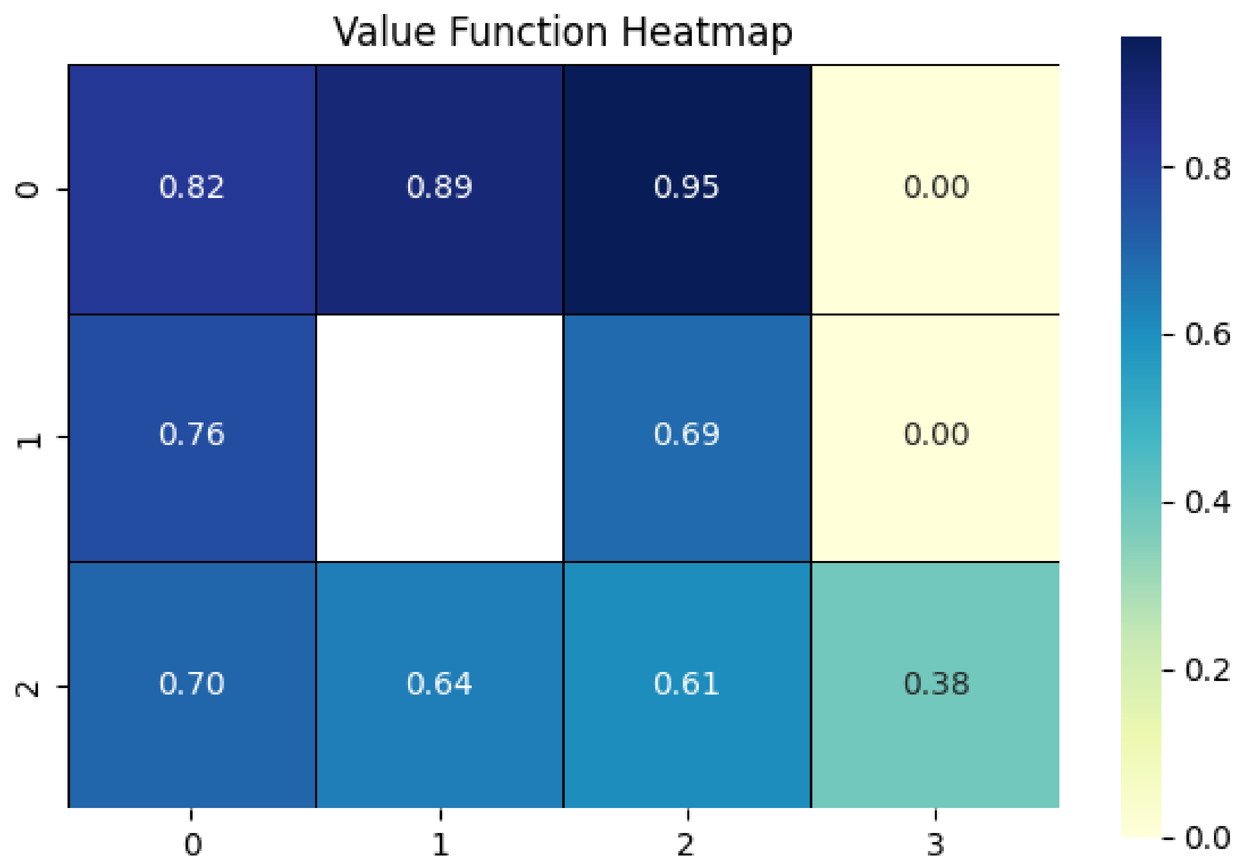
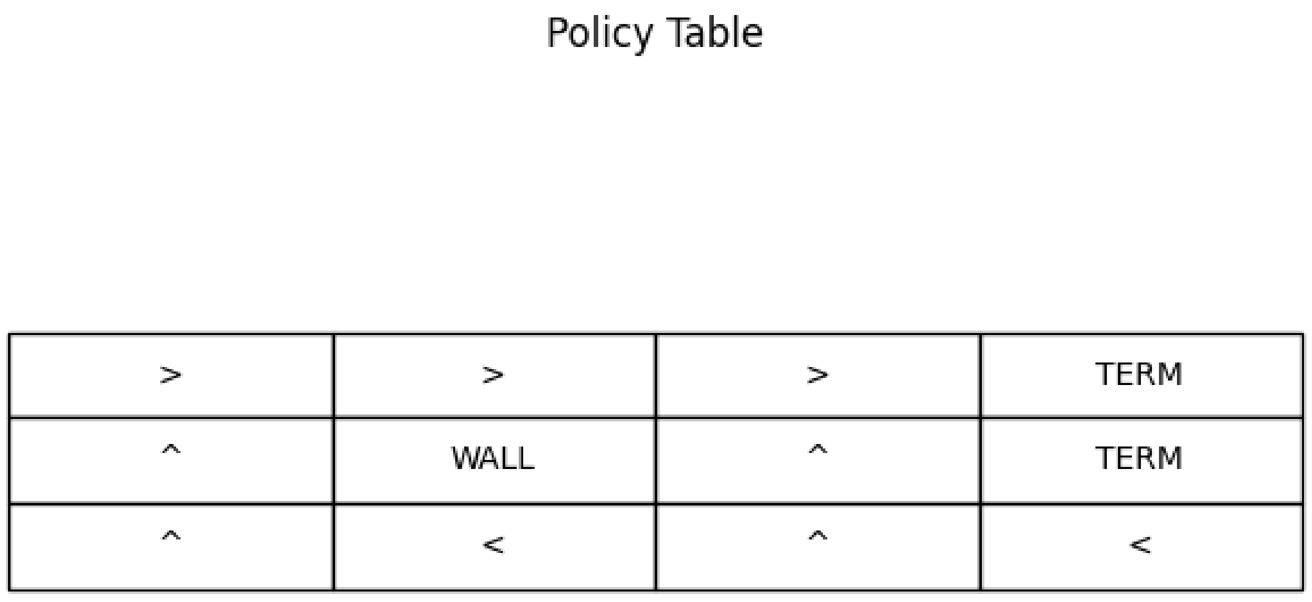
MDP Experiment Report

```
Task 2: Value Iteration Loop
def value_iteration(states, actions, rewards, gamma, theta=0.0001):
    V = {s:0.0 for s in states}
    while True:
        delta = 0
        new_V = V.copy()
        for s in states:
            if s in terminal_states:
                continue
            action_values = []
            for a in actions:
                total = sum(
                    p*(rewards[next_s]+gamma*V[next_s])
                    for p,next_s in get_next_states(s,a))
                action_values.append(total)
            best = max(action_values)
            delta = max(delta, abs(V[s]-best))
            new_V[s]=best
        V=new_V
        if delta<theta:
            break
    return V

Task 3: Policy Extraction
def one_step_lookahead(state,V,rewards):
    return {a:sum(
        p*(rewards[next_s]+gamma*V[next_s])
        for p,next_s in get_next_states(state,a)) for a in actions}

def extract_policy(V,rewards):
    policy = {}
    for s in states:
        if s in terminal_states or s==wall:
            policy[s]=None
        else:
            Q = one_step_lookahead(s,V,rewards)
            policy[s] = max(Q,key=Q.get)
    return policy
```

Final Value Function (Heatmap)



Optimal Policy Table


Policy Table

>	>	>	TERM
^	WALL	^	TERM
^	<	^	<

Analysis of Living Penalty Effects

1. Default Living Penalty (-0.04):
   The agent receives a small negative reward per step.
   The policy successfully avoids the pit at (1,3) and finds the goal at (0,3).

2. Living Penalty = 0.0:
   Without a penalty per step, the agent is less incentivized to take the shortest path.
   The policy still reaches the goal while avoiding the pit, but may not prioritize the fastest path.

3. High Living Penalty (-0.5):
   With a high step penalty, the agent strongly prefers the shortest path to the goal.
   The policy avoids all unnecessary moves and reaches the goal in minimum steps.