Programming Assignment 2
Karan Agarwalla
180050045

Task1:

Transition probabilities and rewards are read into a NumPy array of size |S|*|A|*|S|.

Value Iteration
Algorithm:

$V_0 = |S|\ sized\ 0\ array$
$Tolerance = 1e-12$
$Repeat$:
    $For\ s\ \in S$:
        $V_{t+1}(s) \leftarrow \max_{a \in A} Q(s,\ a,\ V_t)$
    $t \leftarrow t + 1$
$Until\ \left|\left|V_{t+1} - V_t\right|\right|_2 > Tolerance$

where $Q(s, a, V_t)$ *is the action value function corresponding to value function* V_t

Howard Policy Iteration
Algorithm:

$policy = |S|\ random\ array\ with\ values\ in\ range\ [0, |A| - 1]$
$Tolerance = 1e-12$
$Repeat$:
    $For\ all\ improvable\ states\ s\ with\ best\ improvable\ action\ a\ and\ Q(s,a) \geq V(s) + Tolerance\ switch\ to\ a$
$Until\ no\ such\ state\ \&\ action\ exist$

Linear Programming
Algorithm:

$Objective: Minimize(\sum_{s \in S} V(s))$
$subject\ to\ \forall s \in S, \forall\ a\ \in A$:
    $V(s) \geq\ \sum_{s' \in S} T(s,a,s')\{R(s,a,s') +\ \gamma V(s')\}$

Assumptions:
In evaluation of value function, the matrix is invertible or there exists a unique solution to the set of Bellman Equations. Also in case of Howard Policy Iteration, it was observed that the policy switches across two such policies due to machine precision. So I have included a tolerance of 1e-12 for an action to be called an improvable action. Similar value of tolerance is used for value iteration.

Observation:
Value Iteration works the fastest followed by Howard Policy Iteration & Linear Programming.

Task2:

The maze is modelled as an MDP with several end states (those with value = 3) and $\gamma = 0.999$. Only those cells are considered as states whose value is not 1, i.e., are either of start state, end state or empty state. Now for end states there are no outgoing transitions. For all other states there are four possible transitions, i.e., East(3), West(2), North(0) & South(1) modelled with probability 1. Now for a
(s, a) pair the destination state can be:

- Walled State(1): A self-loop with reward -10*n*m where n*m is dimensions of maze
- End State(3): A edge with reward 10*n*m where n*m is dimensions of maze
- Empty State(0)/Start State(2): A edge with reward -2

High negative reward on self-loops discourages such transitions. Negative reward of -2 discourages larger paths. A positive end state reward helps in faster convergence.
Run the algorithm using **Value Iteration.**

Gamma was taken to be 0.999. One can easily see that the value function decreases with increase in number of steps to the end state. However, with $\gamma = 1$ there was an issue when there was a loop of 0's with no end state in sight. In such cases value iteration doesn't converge as the loop is continuous in nature. Hence a value of 0.999 was taken. Also $0.999^{10000} = 4.51 * 10^{-5}$ hence there won't be precision issues even.