Task 1 MDP Planning algorithms

Value iteration algorithm:

```
$V_0 \leftarrow$ Arbitrary, element-wise bounded, $n$-length vector.
$t \leftarrow 0$.
Repeat:
    For $s \in S$:
        $V_{t+1}(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s, a, s')(R(s, a, s') + \gamma V_t(s'))$.
        $t \leftarrow t + 1$.
Until $V_t \approx V_{t-1}$ (up to machine precision).
```

Design decision: V has been stored as a one dimensional array whereas T and S are three dimensional arrays with dimensions of numStates, numActions, numStates and all operations at each iteration are vectorised.
Assumption: A tolerance value of 1e-6 has been used for determining convergence. The initial value function has been chosen to be the vector of all zeros

Howard's policy iteration algorithm:

```
Given $\pi$,
    Pick one or more improvable states, and in them,
    Switch to an arbitrary improving action.
Let the resulting policy be $\pi'$.
```

The above algorithm is the general policy iteration algorithm. In Howard's Policy iteration algorithm all states having improvable actions are improved.
Design decision: V, T and R have been stored as described in value iteration. A for loop over the states has been used to determine the improvable states given a policy.
Assumption: If a state has multiple improvable actions, then the one with the least index has been chosen as the improved action. The initial policy has been chosen to be the policy in which the action 0 is taken from each state.

Linear Programming:

$$\text{Maximise} \left(-\sum_{s \in S} V(s) \right)$$

subject to

$$V(s) \geq \sum_{s' \in S} T(s, a, s')\{R(s, a, s') + \gamma V(s')\}, \forall s \in S, a \in A.$$

Design Decision: Three for loops have been used to formulate the lpp. As a result, the complexity is $\theta(sumStates * numStates * numActions)$. One for loop over s, another over a and finally a for loop for the summation over s'.
Observation: The time taken to formulate the lpp is way larger than the time taken to solve the lpp by the puLP library.

Task 2 Solving a maze using MDPs

States: the naming of states has been done as follows:
The position at row i and column j of the grid is assigned state number of i*cols + j where cols is the total number of columns.

Transition matrix: For each of the states with labels 0 or 2, and a given action, transitions have been added as follows:
If on taking the action:
we reach a 0 block, then the next state is that block
We reach a 1 block(wall), then the next state is the present state
The transition probability has been assigned a value of exactly one for each of the transitions as this is a deterministic process.

Reward: The reward has been chosen to be zero for each transition except the transition where the next state is the end state. In that case, the reward has been chosen to be a large number 1e6 in this case.

Discount factor(γ): the discount factor has been chosen to be 0.9. It can in theory be chosen to be any number close to one but not one. The reason being, if the agent takes t steps to reach the end of the maze, then the total reward will be γt * 1e6 (gamma raised to t). Thus if gamma is 1 then the reward is the same irrespective of the number of steps taken to reach the end. whereas , if gamma is less than 1 then longer time steps receive lesser total reward and would not be optimal.