Report of Team 97

up:
If the input was null, then the output is also null else if the robot's cell has the first coordinate = 0, the output will be null as the robot doesn't go out of bounds. Else we will create a new state for the robot with the first coordinate decremented by 1 and the string of this state will be "up" and the list of cells of this state will be the same as the previous state and the parent state of this state will be the input state.

down:
If the input was null, then the output is also null else if the robot's cell has the first coordinate = 3, the output will be null as the robot doesn't go out of bounds. Else we will create a new state for the robot with the first coordinate incremented by 1 and the string of this state will be "down" and the list of cells of this state will be the same as the previous state and the parent state of this state will be

the input state.

left:
If the input was null, then the output is also null else if the robot's cell has the second coordinate = 0, the output will be null as the robot doesn't go out of bounds. Else we will create a new state for the robot with the second coordinate decremented by 1 and the string of this state will be "left" and the list of cells of this state will be the same as the previous state and the parent state of this state will be the input state.

right:
If the input was null, then the output is also null else if the robot's cell has the second coordinate = 3, the output will be null as the robot doesn't go out of bounds. Else we will create a new state for the robot with the second coordinate incremented by 1 and the string of this state will be "right" and the list of cells of this state will be the same as the previous state and the parent state of this

state will be the input state.

collect:
If the input was null, then the output is also null else we will check if the list of cells contains a cell that is the same as the cell of the robot. If it contains then we will call a helper method collectH else we will return null.

collectH:
takes as an input MyState and outputs also MyState. It creates a new state for the robot and the string of this state will be “collect” and the list of cells of this state will be without the cell that is the same as the robot’s cell. The robot’s cell will not be changed and the parent state of this state will be the input state.

isEqual:
A helper method for collectH. It takes two cells as input and checks if they are the same. It returns true if they are the same else

it returns false.

isNotEqual:
It’s a helper function for collect and it’s the negation of isEqual.

nextMyStates:
If the input is null the output will be an empty list else we will output the list that its members are: applying the up function to the input, applying the down function to the input, applying the left function to the input, applying the right function to the input, applying the collect function to the input. We will filter this list to remove any state that is null.

isNotNull:
It’s a helper for the function nextMyStates and takes as input my state. If the input is null then the output is false, else true.

isGoal:
If the input is null, the output is false. Else if

the length of the list of cells is 0, we return true else false.

search:
If the input is an empty list, the output is null. Else if the head of the list is a goal, the head will be returned. Else we will search again on the list that is the concatenation of the input list's tail and the next my states of the head.

constructSolution:
If the input is null, the output is an empty list. Else if the string of the state is not an empty string, we will concatenate the string to constructSolution of the previous state. Else we will return an empty list.

solve:
If the list is empty, the output is an empty list. Else we will create a myState with the first parameter as the input cell and the second parameter as the input list, the third parameter as an empty string and the fourth parameter as null. We will get the next

myStates of this state which will return a list then we will apply the search function on this list to get an output myState. We will apply the function constructSolution on this myState to give us the desired list of strings.