**German University in Cairo**

**Department of Computer Science**

**Dr. Nourhan Ehab**

***CSEN403 Concepts of Programing Languages***

***Project 2’s report Minesweeper Robot***

***Team 112***

Table

Description automatically generated

***Ahmed Haitham Ibrahim 52-25275***

***Omar Hany Elshafey 52-3681***

***Noureldin Ashraf 52-6290***

***Omar Rafee 52-2676***

Text

Description automatically generatedFirst, we start by defining the cell type which is a position on the grid pair coordinate representing

a row number(x), and a column number(y).

Then we define the 4 methods of the movements which they take as an input a MyState type and return a MyState type of data, and before we do that we should check that the state of the robot is not going to move out of the 4x4 grid.

Up: takes a state and checks whether the x coordinate is bigger than 0 and if it’s bigger than 0 it increments the x coordinate in the state.

Down: takes a state and checks whether the x coordinate is smaller than 3 and if it’s smaller than 3 it decrements the x coordinate in the state.

Left: takes a state and checks whether the y coordinate is bigger than 0 of it’s bigger than 0 it decrements the y coordinate in the state.

Right: takes a state and checks whether the y coordinate is smaller than 3 and if it’s smaller than 0 it increments the y coordinate in the state.

Graphical user interface, text

Description automatically generated with medium confidence

We will define 2 helper methods find a [], remove a[].

Find a [] : takes as an input a list and checks whether the item we want to find (a) is in the list or not it does that by checking the head of the list and if it’s equal to a it returns true if not it calls itself again recursively on the other elements.

Remove a []: takes as an input a list and checks whether the item we want to remove (a) is in the list by comparing a with the head of the list and It removes the item by returning the rest of the list.

After that we should be able to implement the method collect which has the role of collecting the mines from the grid.

Collect: takes as an input a Mystate data type and returns a Mystate data type, it collects the mine from the input state and if it’s collected it removes the mine from the list of mines by calling remove a[]

If there’s no mine on the current cell it returns null.

Now in order to implement nextMyStates we should define a helper method called clean

What clean does is that it checks the head of the any list and if it’s equal to null it returns the rest of the list. Here we implemented it recursively so that we clean the whole list by calling this method we should return a non-Null list.

NextMyStates: the method takes as an input a Mystate data type and returns a list of Mystate data type

This method represents all the possible actions that the robot can take to collect a mine and by calling such methods like right, left, up, down and collect can result in a null result and here comes the role of clean which will take all null result and remove if from the resulting list.

Text

Description automatically generated

IsGoal: takes as an input a Mystate data type and returns false if the list of mines has no mines and true if the list still has mines.

Search: it takes a list of Mystate as an input and returns only one Mystate and what this method does is checking whether the head of the list has mines or not if it has mines it returns it if not it calls itself recursively on the rest of the list.

ConstructionSolution: it takes as an input a Mystate data type and returns a list of strings which will represent the actions that the robot can do to collect all mines in the state.

Solve: is where the game actually starts we give the method the robot cell and a list of cells of the mines on the grid and we expect it to return us a string containing the possible action to be done to collect them all. And it does this by calling search on the input and we call constructionSolution to make sure that it returns us a string that we can read.