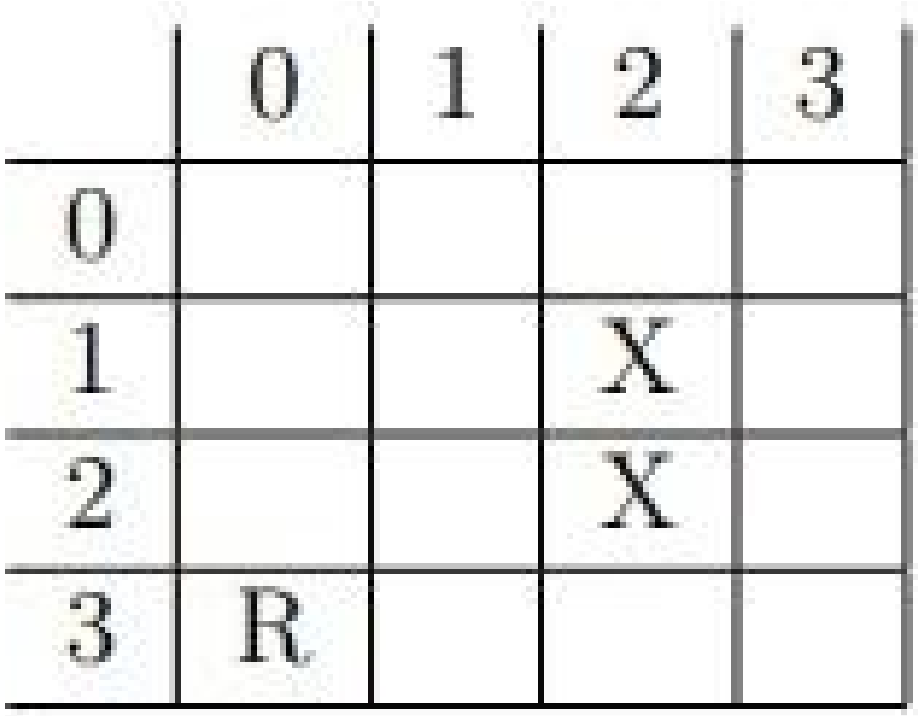
Concepts of Programming Languages

Project 2

Team 86

Ahmed Hussein 52-13398
Lujina Amer 52-6599
Mohamed Ahmed 52-27145
Ahmed Mohsen 52-22728



Functions :

1. Movement functions : Functions responsible for moving the robot
    a. up:: MyState -> MyState The function takes as input a state and returns the state resulting from moving up from the input state. If up will result in going out of the boundaries of the grid, Null should be returned.
    b. down:: MyState -> MyState The function takes as input a state and returns the state resulting from moving down from the input state. If down will result in going out of the boundaries of the grid, Null should be returned.
    c. left:: MyState -> MyState The function takes as input a state and returns the state resulting from moving left from the input state. If left will result in going out of the boundaries of the grid, Null should be returned.
    d. right:: MyState -> MyState The function takes as input a state and returns the state resulting from moving right from the input state. If right will result in going out of the boundaries of the grid, Null should be returned.
2. collect:: MyState -> MyState

    The function takes as input a state and returns the state resulting from collecting from the input state. Collecting should not change the position of the robot, but removes

the collected mine from the list of mines to be collected. If the robot is not in the same position as one of the mines, Null should be returned.

3. nextMyStates :: MyState->[MyState] : A function that takes current state and returns all the possible movements for the robot in that position including collecting mines (uses the 5 functions mentioned above).
4. search::[MyState]->MyState : A function that searches for the goal state that has [] (no mines left to collect) which finishes the task of collecting mines and returns the goal state.
5. constructSolution:: MyState ->[String] : A function that gets the sequence of movements and collects from the current state and returns them in a List of Strings.
6. solve :: Cell->[Cell]->[String] : A function that uses the search and constructSolution functions to return a valid solution to collect all the mines on the grid (basically the goal of the project).

Runs :

1.Code run :

```
Type :? for help
Hugs> :load "C:\\Users\\ahmed\\Desktop\\guc\\cs403\\cs403 project\\Project2.hs"
Main> solve (3,0) [(2,2),(1,2)]
["up","right","right","collect","up","collect"]
Main> solve (3,0) [(1,1),(0,0)]
["up","up","up","collect","down","right","collect"]
Main>
```

2. Bonus runs : we adjusted the program for a 10x10 grid to be easy for tracing.

```
Main> solve (9,0) [(0,0),(7,7),(1,0),(1,5),(4,4)]
["up","up","up","up","up","up","up","up","collect","up","collect","down","right","right","right","right","right","collect","down","down","down","left","collect","down","down","down","right"
,"right","right","collect"]
Main> solve (9,0) [(0,0),(7,7),(1,0),(1,5),(4,4),(7,7)]
["up","up","up","up","up","up","up","up","collect","up","collect","down","right","right","right","right","right","collect","down","down","down","left","collect","down","down","down","right"
,"right","right","collect","collect"]
```

	0	1	2	3	4	5	6	7	8	9
0	X	X								
1										
2										
3										
4					X					
5		X								
6										
7								X		
8								X		
9	R									

But it can be adjusted to higher number even more than 100 :

Procedures:

1. We added 5 new functions :
    a. distance : A function that gets the distance between two cells using Manhattan distance.
    b. nearest : A function that gets the nearest mine to the current state of the robot.
    c. movev : A function that moves in a vertical axis.
    d. moveh : A function that moves in a horizontal axis.
    e. move : A function that uses movev and moveh to move in both axes to the nearest mine.
2. We modified the search function to move to each mine one after the other using the move function and collect each of them, after each collect it checks if the current state is the goal state(no mines left to collect).