Concepts of Programming Languages (CSEN403) - Spring 2022
Project, Milestone 2
Team 32

REPORT

Milestone 2

Team 32

Abdelrahman Mohamed Ahmed Abouelkheir *(52-5388)*
Ahmed Sherif Said Ali El Sayed *(52-8068)*
Ahmed Ayman Ahmed Gomaa *(52-18268)*
Mahmoud Mohammed Mahmoud Abou Eleneen *(52-5514)*

June 2022

Introduction

The following is a short report containing:

1. A copy of our Haskell code for milestone 2 of the project for [CSEN403] for the Spring Semester of 2022,
2. A brief description of the implemented functions,
3. Screenshots of two different grid configurations with the returned solutions.

1. Code

The following is a copy of our Haskell code submitted in the file “Team 32.hs” alongside this report; displayed here for ease of viewing.

```
-- Type & data declarations
type Cell = (Int,Int)
data MyState = Null | S Cell [Cell] String MyState deriving (Show,Eq)

-- Helper Functions

--(delete item from list)
delItem _ [] = []
delItem x (y:ys) | x == y = delItem x ys
                 | otherwise = y : delItem x ys

--(use in nextMyStates to check if applying up/down/right/left returns null or not)
produceNextStates f        | f == Null = []
                                        | otherwise = [f]

-- Main Code

--up
up :: MyState -> MyState
up Null = Null
up (S (x,y) list string prevstate) | x>0 = (S (x-1,y) list "up" (S (x,y) list string
prevstate) )
                                                                | otherwise = Null
```

```
--down
down :: MyState -> MyState
down Null = Null
down (S (x,y) list string prevstate) | x<3 = (S (x+1,y) list "down" (S (x,y) list string
prevstate) )
                                                    | otherwise = Null

--left
left :: MyState -> MyState
left Null = Null
left (S (x,y) list string prevstate) | y>0 = (S (x,y-1) list "left" (S (x,y) list string
prevstate) )
                                                    | otherwise = Null

--right
right :: MyState -> MyState
right Null = Null
right (S (x,y) list string prevstate) | y<3 = (S (x,y+1) list "right" (S (x,y) list string
prevstate) )
                                                     | otherwise = Null

--collect
collect :: MyState -> MyState
collect Null = Null
collect (S (x,y) list string prevstate) | elem (x,y) list = (S (x,y) (delItem (x,y) list)
"collect" (S (x,y) list string prevstate) )
                                                          | otherwise = Null

--nextMyStates
nextMyStates :: MyState -> [MyState]
nextMyStates Null = []
nextMyStates inpState = produceNextStates (up inpState) ++ produceNextStates
(down inpState) ++ produceNextStates (left inpState) ++ produceNextStates (right
inpState) ++ produceNextStates (collect inpState)
```

```
--isGoal
isGoal :: MyState -> Bool
isGoal Null = False
isGoal (S (x,y) list string prevstate) | (length list == 0) = True
                                                              | otherwise = False

--search
search :: [MyState] -> MyState
search [] = Null
search (y:ys) | isGoal y = y
                          | otherwise = search ( ys ++ (nextMyStates y) )

--constructSolution
constructSolution state = reverse(h state)
h (S (y,x) l s p) | s=="" = []
                  | otherwise = s:h p

--solve
solve :: Cell -> [Cell] -> [String]
solve (x,y) l = constructSolution (search  [(S (x,y) l "" Null)])
```

2. Description of implemented functions

The following is a brief description of the implemented (helper & main) functions in our Haskell code.

--(delete item from list)
- - - > i made this helper function to delet the mine after i collect it from the board

(produceNextStates)
- - - - >This method used to check what move will i apply

(up)
This method used to done the the move up of the robot

(down)

This method used to done the move down of the robot

(left)

This method used to done the move left of the robot

(right)

This method used to done the move right of the robot

(collect)

This function takes my state as an input and gives me my state after collecting the mine , also i used the helper function i made which is deleteitem to delete the mine from the board

(nextmystates)

This function takes mysate and gives me mystate after doing any move , also i used the helper function produce next states to clarify which move i will do

(isGoal)
This method takes mystate as an input and gives me true or false while checking If there are more mines to be collected or not

(search)
This function briefly checking if the head of the list is a goal or not .. if not it calls itself again till reaching the tail

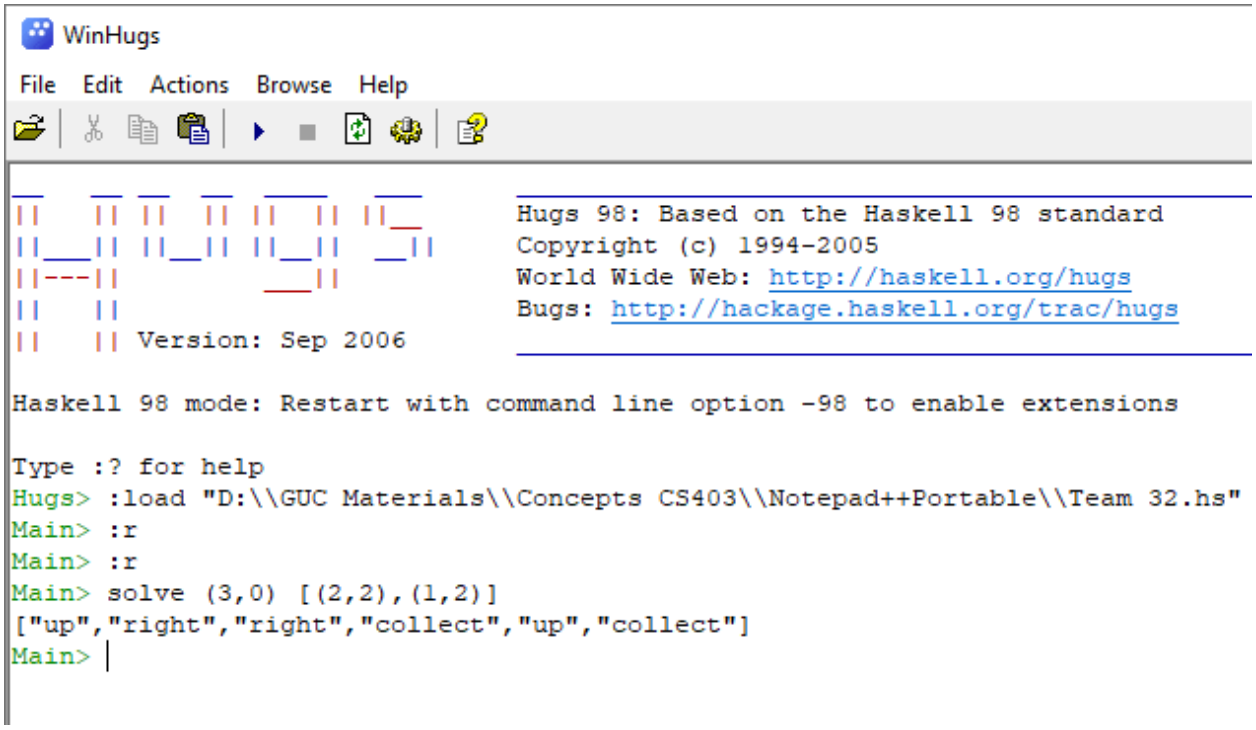
(constructionsolution)
This method takes the state and give me the list of instructions i have to do reaching to the mine

(Solve)

The function takes as input a cell starting position of the robot, a cells the positions of the mines, and returns a set of strings representing actions that the robot can follow to reach a goal state from the initial state.

3. Screenshots & returned solutions

The following are screenshots of two different grid configurations with the returned solutions.



```
Main> :r
Main>  solve (3,0) [(3,2),(2,0)]
["up","collect","down","right","right","collect"]
Main>
```

And now this my bonus screenshots

>

```
Type :? for help
Hugs> :load "C:\\Users\\Ahmed Sherif\\Desktop\\Bonus team 32....).hs"
Main> :r
Main> solve (3,0) [(2,2),(1,2),(1,4)]
["up","right","right","collect","up","collect","right","right","collect"]
Main> solve (3,0) [(2,2),(1,2),(1,3)]
["up","right","right","collect","up","collect","right","collect"]
Main>
```