**UNIX PROJECT**

Course Code: IT200



**Submitted to:**

Ms.Deepthi

Assistant Lecturer

Department of Information Technology

NITK Surathkal

Ms.Sangeeta

Assistant Lecturer

Department of Information Technology

NITK Surathkal

Submitted by:

SACHIN – 15I234

KANDUKURI ABHISHEK – 15IT214

CH RAMPRAKASH – 15IT115

11th November 2016 ABHISHEK S – 15IT202

***CONTENTS:***

Cover page ……………………………………………………………….. 1 – 2

Contents …………………………………………………………………. 3

Certificate ………………………………………………………………… 4

Abstract ………………………………………………………………….. 5

Introduction …………………………………………………………………. 6

Code …………………………………………………………………. 7 – 22

Output ………………………………………………………………… 24 – 26

Algorithm ………………………………………………………………… 27 – 28

Conclusion ………………………………………………………………… 29

Certificate:



This is to certify that the Btech Mini project entitled “**UNIX MINI** **PROJECT”** submitted by :

**SACHIN – 15I234**

**KANDUKURI ABHISHEK – 15IT214**

**CH RAMPRAKASH – 15IT115**

**ABHISHEK S – 15IT202**

as the record of the work carried out by then is accepted as the Btech Mini project submission in partial fulfillment of the requirements for mandatory learning course of IT202, Unix Programming and Practice in the Department of Information Techonlogy.

Date: 15th November 2016.

Place: Surathkal.

Signature and Seal

Abstract:



The game is **MINESWEEPER GAME** and is made using BASH Shell Scripting.

Here the code is divided in 3 parts:-

* **Drawing Boards and Putting of Mines:-** Using 2-D array and Random function we will create board and place mines at random position,
* **Input from user and Searching:-** Input from user is taken and corresponding element is searched.
* **Deciding: -** we check whether the respective player satisfiesthe winning condition or not.

The law of the game include taking the input coordinate from the keyboard entered by player and then corresponding board cell will be raveled and if the player is the hitted then game will be terminated and if all till end player doesn’t hit the mine then at end the player will won.

Introduction:



This game is a take on the classic game MineSweeper. The game places the player in minefield where

the objective is to avoid all the mines laid out under the tiles. The mines are triggered when the

player uncovers them. The mines are never in the same place,this ensures that every game is a new

experience. This is strategy game where the player has to find the patterns and use logical

reasoning to get himself out of this mine field. The game operates on a co-ordinate input system.

The only win scenario int this game is when the player uncovers all the tiles except the ones

covering the mines and the player will be awarded his freedom from the minefield.

The game uses shell script and uses many of the useful commands such as shuf command, the command

used to give a random number in a given range and declare command, the command used to declare a

2d array in shell.Some interesting features of shell have been used like changing the color code

to display different colors on the terminal. The goal of the project is to gain proficiency in

shell scripting,hence the project was coded in shell script.

.code

#!/bin/bash

clear

n=10

m=10

mine=15

mine2=$mine

i=0

j=0

k=0

r=0

declare -A board

declare -A fore

for((i=0;i<n;i++))

do

for((j=0;j<m;j++))

do

board[$i,$j]=0 // board and for both r zero

fore[$i,$j]=0

done

done

i=0

j=0

d=0

drawBoard() #creating

{

#place mines

for((i=0;i<n;i++))

do

for((j=0;j<m;j++))

do

r=$(shuf -i 1-100 -n 1) #// placing mines at random position ...used random fn

if ((board[$i,$j] !=-1))

then

if ((r<=5&&mine!=0)) # mines are given a probability 5% and the program cycles till 15 mines are placed

then # placing mine at that positon coordu=i nate

board[$i,$j]=-1

((mine--))

else

board[$i,$j]=0 # no mines

fi

fi

if ((i==n-1&&j==m-1&&mine!=0)) # if all the mines havent be placed then again ititating the loop -1 is for mines

then

i=0

j=0

fi

done

done

#update board #for calculating number of mines in the surroundings

for((i=0;i<n;i++))

do

for((j=0;j<m;j++))

do

if ((board[$i,$j] == -1)) # going to the mine and udate all 8 boxex around it and every position show that how many mines are there around it

then

for((x=-1;x<=1;x++))

do

for((y=-1;y<=1;y++))

do

d=$((i+x))

e=$((j+y))

if ((d>-1 && d<n && e>-1 && e<m && board[$d,$e] != -1))

then

((board[$d,$e]++)) #if there is mine increase the number in its 8 neighbours

fi

done

done

fi

done

done

}

alpha() #display for testing

{

clear

echo

echo -e "\t\t\t\t \033[32m$n X $m GRID ----- $mine2 MINEs\033[0m \n"

echo

echo -ne " \t\t\t \033[33mcol\033[0m\t "

for((i=0;i<m;i++))

do

echo -ne " \033[33m$i\033[0m " #echo -ne display in the same line

done

echo

echo

for((i=0;i<n;i++))

do

echo -ne "\t\t\t\033[33mrow $i\033[0m \t "

for((j=0;j<m;j++))

do

if((board[$i,$j]==0))

then

/bin/echo -ne "\e[0;37m . \e[0m "

elif((board[$i,$j]==1))

then

/bin/echo -ne "\e[1;34m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==2))

then

/bin/echo -ne "\e[1;32m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==3))

then

/bin/echo -ne "\e[1;35m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==4))

then

/bin/echo -ne "\e[1;33m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==5))

then

/bin/echo -ne "\e[0;34m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==6))

then

/bin/echo -ne "\e[0;32m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==7))

then

/bin/echo -ne "\e[0;35m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==8))

then

/bin/echo -ne "\e[0;33m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==-1))

then

/bin/echo -ne " \e[36;7m#\e[0m "

fi

done

echo

echo

done

echo

}

display() #displays board

{

#display-actual

clear

echo

echo -e "\t\t\t\t \033[32m$n X $m GRID ----- $mine2 MINEs\033[0m \n"

echo

echo -ne " \t\t\t \033[33mcol\033[0m\t "

for((i=0;i<m;i++))

do

echo -ne " \033[33m$i\033[0m "

done

echo

echo

for((i=0;i<n;i++))

do

echo -ne "\t\t\t\033[33mrow $i\033[0m \t "

for((j=0;j<m;j++))

do

if ((fore[$i,$j]==1)) #intiallly fall fore part is zero , e=whenever we put x, y coordi it will become one and correspondoitng position position in thee backhand booard will be revealed

then

if((board[$i,$j]==0))

then

/bin/echo -ne "\e[0;37m . \e[0m "

elif((board[$i,$j]==1))

then

/bin/echo -ne "\e[1;34m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==2))

then

/bin/echo -ne "\e[1;32m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==3))

then

/bin/echo -ne "\e[1;35m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==4))

then

/bin/echo -ne "\e[1;33m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==5))

then

/bin/echo -ne "\e[0;34m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==6))

then

/bin/echo -ne "\e[0;32m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==7))

then

/bin/echo -ne "\e[0;35m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==8))

then

/bin/echo -ne "\e[0;33m ${board[$i,$j]} \e[0m "

elif((board[$i,$j]==-1))

then

/bin/echo -ne "\e[1;31m \* \e[0m "

fi

else

echo -ne " # "

fi

done

echo

echo

done

echo

}

refresh() # reset game variables

{

for((i=0;i<n;i++))

do

for((j=0;j<m;j++))

do #again gavme will start

board[$i,$j]=0

fore[$i,$j]=0

done

done

n=10

m=10

mine=15

i=0

j=0

k=0

r=0

x=0

y=0

d=0

e=0

}

gameover()

{

for((i=0;i<n;i++))

do

for((j=0;j<m;j++))

do

if ((board[$i,$j] == -1))

then

fore[$i,$j]=1

fi # when you hit a mine then itw will revela is

done

done

display

echo -e "\e[0;31m \t\t\t\t All the mines have exploded. \e[0m"

echo -e "\e[0;31m \t\t\t You could not survive the explosion. \e[0m"

echo -e "\e[0;31m \t\t\t\t\t Game Over \e[0m\n\n\n\n\n\n\n"

echo -e "\e[0;36m \t\t\t\t Press [ ENTER ] to continue. \e[0m\n\n"

read

}

checkwin() # checks win at every move

{

win=0

for((q1=0;q1<n;q1++))

do

for((q2=0;q2<m;q2++))

do

if ((fore[$q1,$q2] == 1))

then

if ((board[$q1,$q2] == -1))

then

gameover

win=999

fi

((win++))

fi

done

done

if ((win==m\*n-mine2)) #if you have uncoverd the whole board except the mines m\*n=total n.o tiles

then

echo -e "\e[0;32m \t\t\t\t You Have Won The Game !!! \e[0m"

echo -e "\e[0;33m \t\t\t\t Avoided all the mines. \e[0m"

echo -e "\e[0;36m \t\t\t\t All mines defused. \e[0m\n\n"

echo -e "\e[0;36m \t\t\t\t Press [ ENTER ] to continue. \e[0m\n\n"

read

i=$((m\*n))

fi

}

backdoor()

{

alpha

echo -e "\e[0;32m \t\t\t\t You Have Won The Game !!! \e[0m"

echo -e "\e[0;33m \t\t\t\t Avoided all the mines. \e[0m"

echo -e "\e[0;36m \t\t\t\t All mines defused. \e[0m\n\n"

echo -e "\e[0;36m \t\t\t\t Press [ ENTER ] to continue. \e[0m\n\n"

read

i=$((m\*n))

}

chain()

{

i=$1

j=$2

fore[$i,$j]=1

if ((board[$i,$j] == 0))

then

reveal $i $j # reveal the neighbours if zero

revzero #checks if any revealed neighbours is zero and then reveals their neighbours if present

fi

}

reveal() # reveals the 8 neighbours of given input

{

i=$1

j=$2

fore[$i,$j]=1

for((x=-1;x<=1;x++))

do

for((y=-1;y<=1;y++))

do

d=$((i+x))

e=$((j+y))

if ((x!=0 || y!=0))

then

if ((d>-1 && d<n && e>-1 && e<m && fore[$d,$e] == 0))

then

fore[$d,$e]=1

fi

fi

done

done

}

revzero() # Scans the board if there are any zeros with covered neighbours and reveals them

{ # then it scans the whole board again.

for((i=0;i<n;i++))

do

for((j=0;j<m;j++))

do

k=1

for((x=-1;x<=1;x++))

do

for((y=-1;y<=1;y++))

do

d=$((i+x))

e=$((j+y))

if ((x!=0 || y!=0))

then

if ((d>-1 && d<n && e>-1 && e<m && fore[$d,$e] == 0))

then

k=0

fi

fi

done

done

if ((fore[$i,$j] == 1 && board[$i,$j] == 0 && k==0))

then

reveal $i $j

i=0

j=0

fi

done

done

}

instructions()

{

fiv="\e[0;34m5\e[0m"

sta="\e[1;31m\*\e[0m"

clear

echo -e "\n\n\t\t\t\t \e[5;32;40m Welcome to MINE MANIA \e[m\n"

echo -e "\n\n\t\e[0;36mInstructions:\e[0m\n"

echo -e "\tYou are a \e[0;33mSoldier\e[0m stranded in a minefield.Your mission is to navigate"

echo -e "\tthrough it without setting off any mines.\n"

echo -e "\tYou know that:"

echo -e "\tThe field is a \e[0;33m10X10 grid\e[0m and there are \e[0;33m15 mines\e[0m.\n"

echo -e "\tThe purpose of the game is to open all the tiles of the field which do not contain"

echo -e "\ta mine. You lose if you set off a mine tile.\n"

echo -e "\tEvery non-mine tile you open will tell you the total number of mines in the eight"

echo -e "\tneighboring tiles.\n"

echo -e "\tFor example,\n"

echo -e "\t # # # $sta # $sta $sta # $sta $sta # #"

echo -e "\t # $fiv # = $sta $fiv $sta or $sta $fiv # or $sta $fiv #"

echo -e "\t # # # # $sta # $sta # $sta $sta $sta $sta"

echo -e "\n"

echo -e "\tOnce you are sure that a tile contains a mine, do not open it and uncover all the"

echo -e "\trest of the tiles.\n"

echo -e "\tEnter the X co-ordinate and Y co-ordinate of the tile to open it.\n"

echo -e "\t\e[31mCAUTION\e[0m: Pressing [Enter] without co-ordinate input results in random selection."

echo -e "\tIf you uncover a mine then the game terminates."

echo -e "\n\n\t\t\t\t \e[5;32;40m Happy Hunting ! \e[m\n\n\n\n"

echo -e "\e[0;36m \t\t\t\t Press [ ENTER ] to continue. \e[0m\n\n"

read

}

newgame() # game procedure

{

refresh

drawBoard

#alpha

display

for((i=0;i<m\*n-mine;i++))

do

x=500

y=500

read -p " X co-ordinate : " x

read -p " Y co-ordinate : " y

if [ "$x" == "mine" -a "$y" == "mania" ]

then

backdoor

#i=$((m\*n))

elif ((x>-1 && x<n && y>-1 && y<m))

then

chain $x $y

if ((board[$x,$y]==-1))

then

gameover

i=$((m\*n))

else

display

fi

else

echo -e "\e[0;31m \t\t\t Invalid Position \e[0m "

fi

checkwin

done

}

rollcredits() # credits

{

clear

echo -e "\n\n\n\n\n\n"

echo -e "\e[0;32m \t\t\t\t See you next time.\e[0m\n"

echo -e "\e[1;34m \t\t\t\t Thank You For Playing !!!\e[0m\n"

echo -e "\n\n\n\n\n\n\n\n\n"

echo -e "\e[1;34m \t\t\t\t\t Game created by:\e[0m\n\n"

echo -e "\e[0;32m \t\t\t\t\t Sachin \e[0m\n"

echo -e "\e[0;34m \t\t\t\t\t Ramprakash \e[0m\n"

echo -e "\e[0;33m \t\t\t\t\t Abhishek K \e[0m\n"

echo -e "\e[0;31m \t\t\t\t\t Abhishek S \e[0m\n\n\n\n\n\n\n\n"

echo -e "\e[0;36m \t\t\t\t Press [ ENTER ] to exit. \e[0m\n\n"

read

echo

clear0

}

#main method starts here

p1='y'

for (( ;(p1 == 'y') || (p1 == 'Y'); ))

do

instructions

newgame

echo -ne "\t\t\t Would you like to play again? (y/n) "

read p1

done

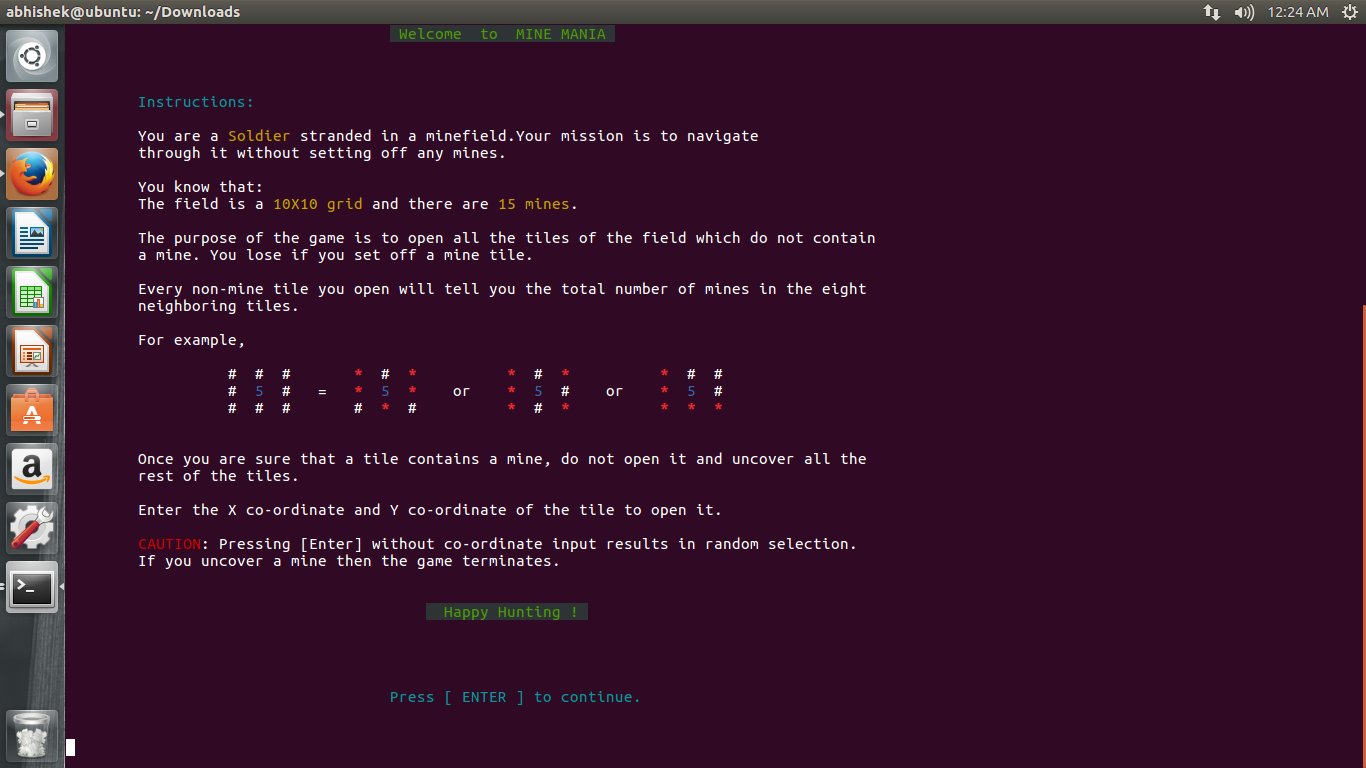
rollcredits

clear

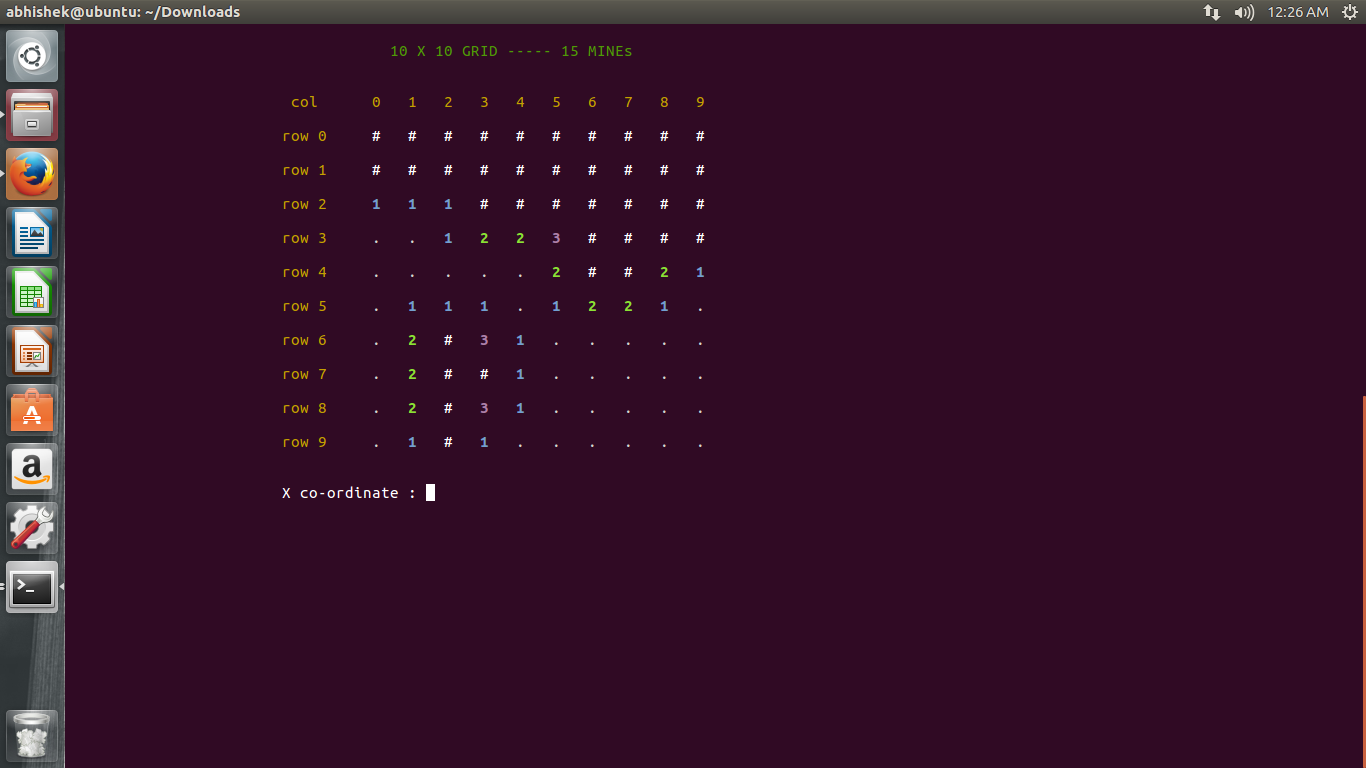
0

OUTPUT:

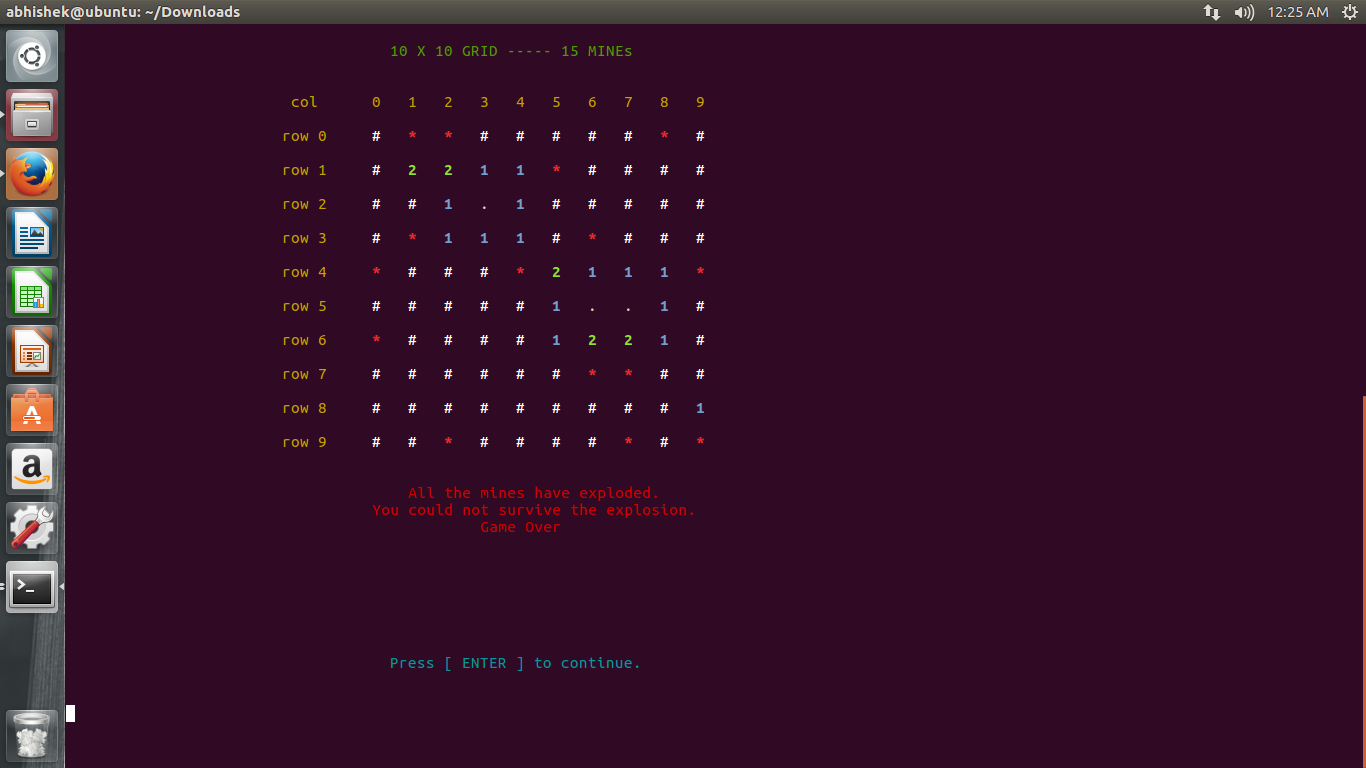
The initial output screen



The intermediate screen where the adjacent zeroes are shown.



Finally the endgame screen



Algorithm:



Step-1:

Our project involve drawing of two board first one is fore board(named as fored) and second one is back board(named as bored) so we use 2-D array for that we used **declare -A** command.

Step-2 :

We now create the board and place the mines in it using the function **DrawBoard().** In this function for placing mines at random position we use **shuf** command with the probability of 5% and we use 15 mines in the whole game. If 15 mines are not placed using random function then the loop again iterate.

Step-3 :

Next we use **Update()** function to update the board after each turn it will search for the mines in surrounding 8 squares if there is a mine then it will increase the mine no of surrounding 8 shell by one.

Step-4 :

Next we use **display()** function to display the board after each turn input coordinate in fore become one and corresponding coordinate in board is revealed using echo –ne command which display in same line and ([1;34m ${board[$i,$j]} ) use to display the color.

Step-5 :

Next in this code we uses the the **refresh() this** function again refresh the board and put mines 15 and make the every cell of fore and board equal to zero.

The **Gameover()** function will terminate the game game . if in the bore cell contain mine corresponding to the input coordinate then it will call the display function and terminate the game.

Step-6 :

**Checkwin()** iterate throughout the board and check for the mine after each turn if mine is there then it will call the **gameover()** function and terminate the game. Else if all cell are uncovered then it display the message that player has won the game.

Step-7 :

This is the main step in the algorithm of the game in this it will take the x, y coordinate as input from the **newgame()** function and check whether the corresponding board coordinate if those are zero then it will call **reveal()** function which reveals the 8 neighbours of given input

If any of 8 input is zero then the chain function will call the revzero() which Scans the board if there are any zeros with covered neighbours and reveals then then it scans the whole board again.

Step-8 :

From main method, if user input **‘y’ or ‘Y’** we call the **instructions()** method which display rules of the game then we call the new game function

In **newgame()** function we will take the input coordinate from the user if x > -1 and y >-1 it will call the **chain()** function if board coordinate is -1 and -1 then it will call **gameover()** method else it will display the result.

CONCLUSION

The game aimed at implementing the classic game MineSweeper using the knowledge gained in the

subject shell scripting.The game is an interactive co-ordinate based game which involves a lot of

logical reasoning.The many commands used int the code are a product of research on the topic as

well as knowledge taught in class. New commands were learnt and implemented in the program such

as shuf command, the command used to give a random number in a given range and declare command,

the command used to declare a 2d array in shell. The usage of functions was seen throughout the

program,the functions played a major part in the program.

The commands used:

clear -Clearing the terminal

echo -Printing various elements

declare -Declaring Arrays

for loops -Various loops

if statements -Various conditions

The functions used:

drawboard() -To setup a new minefield

alpha() -To test the program

display() -To display the board after each move

refresh() -To reset game variables to initial values

gameover() -Display losing Scenario

checkwin() -Display winning Scenario

backdoor() -Alternate to winning Screen (mine mania)

chain() -Initiates the changes to the board after each move

reveal() -Reveals the surrounding 8 neighbours of given tile

revzero() -Reveals the chain of zeroes and their neighbours

instructions() -Instructions screen

newgame() -To begin a new game

rollcredits() -End screen

References:

https://linux.die.net/man/1/shuf

https://linux.die.net/abs-guide/declareref.html

THANK YOU