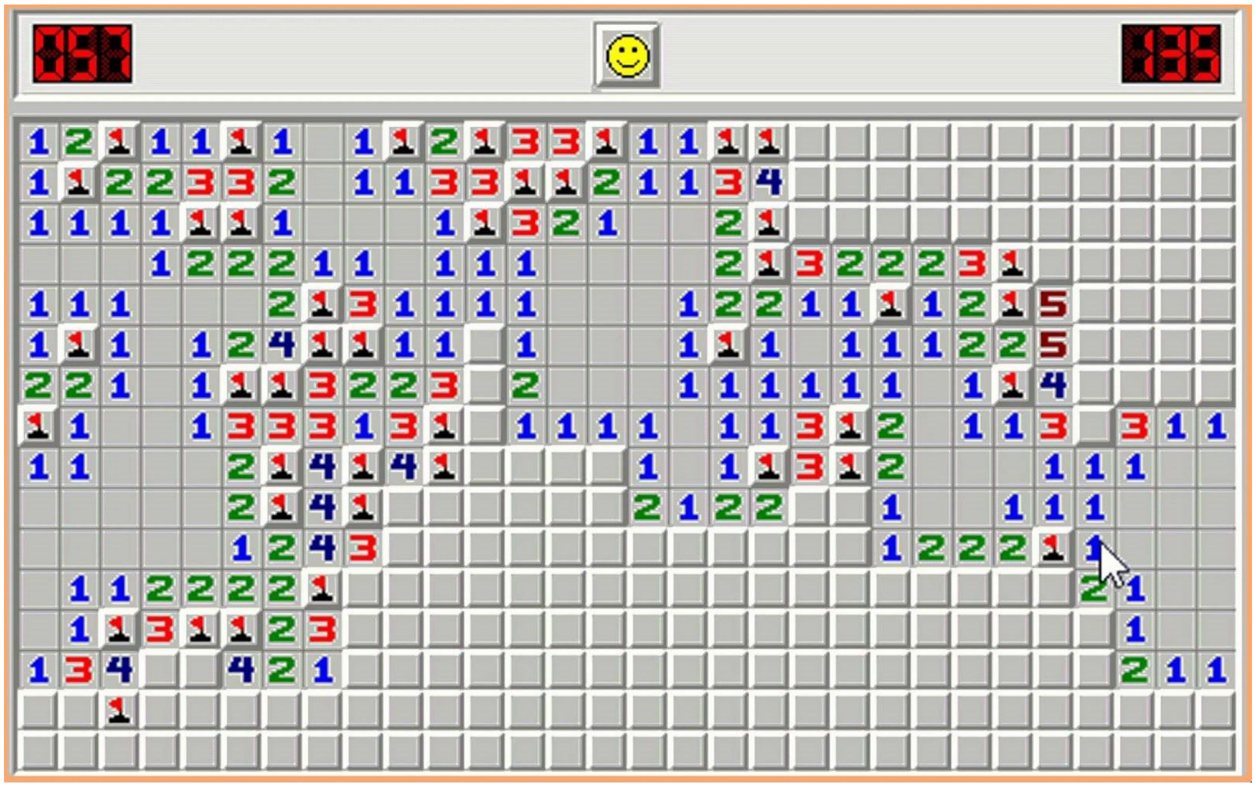
Minesweeper game

Using C programming language



By D/Ashraf's students:

Moatassem Desouky 4497

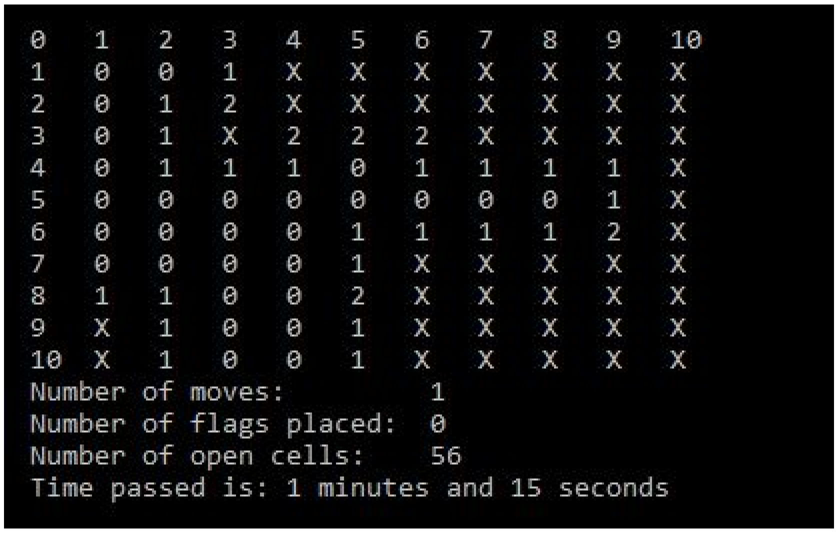
Mustafa Ahmed Ibrahim 4435

3rd term CCE

INTRODUCTION

Minesweeper is a single-player puzzle video game. It has its origins in the earliest mainframe games of the 1960s and 1970s. The objective of the game is to clear a rectangular board containing hidden mines without detonating any of them, with help from clues about the number of neighboring mines in each field. The game originates from the 1960s, and has been written for many computing platforms in use today. In this project we made our own version of the game, figure (1-1) , using C programming.

Our vision was not only to recreate the game with our own algorithms, but also to achieve a bugless, stable game that will hold up even against malicious users.



```
0   1   2   3   4   5   6   7   8   9   10
1   0   0   1   X   X   X   X   X   X   X
2   0   1   2   X   X   X   X   X   X   X
3   0   1   X   2   2   2   X   X   X   X
4   0   1   1   1   0   1   1   1   1   X
5   0   0   0   0   0   0   0   0   1   X
6   0   0   0   0   1   1   1   1   2   X
7   0   0   0   0   1   X   X   X   X   X
8   1   1   0   0   2   X   X   X   X   X
9   X   1   0   0   1   X   X   X   X   X
10  X   1   0   0   1   X   X   X   X   X
Number of moves:         1
Number of flags placed:  0
Number of open cells:    56
Time passed is: 1 minutes and 15 seconds
```

(Figure 1-1)

Game features:

The game allows you to:

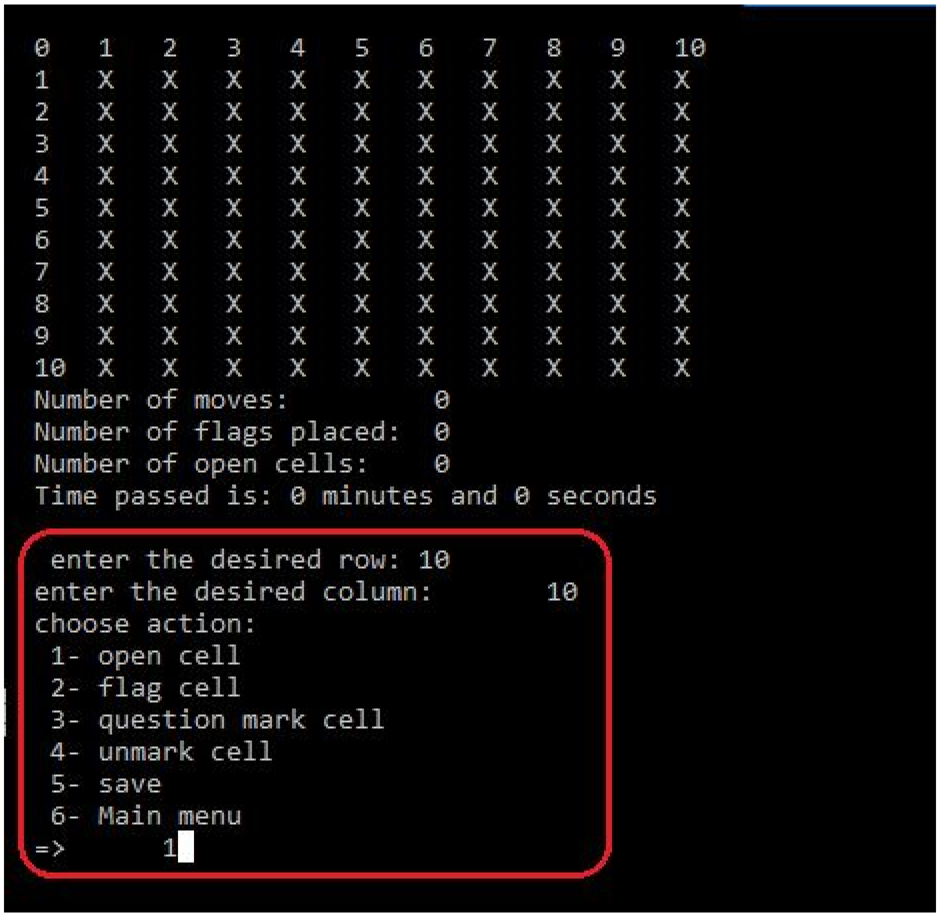
- choose a custom size for the puzzle to match your desired difficulty.
- Check your score and rank amongst other player
- Save the current game progress for later use and load it when needed.
- Check the number of moves so far
- The numbers of flags placed
- And the time passed since the start of the game

Gameplay:

The game guide you through playing it. Anyone in contact with the game for the first time would be able to run it as it gives you options to choose your next step from, as shown in figure (2-1) and the modification made based on the choice is shown in figure (2-2).

During the game itself you can:

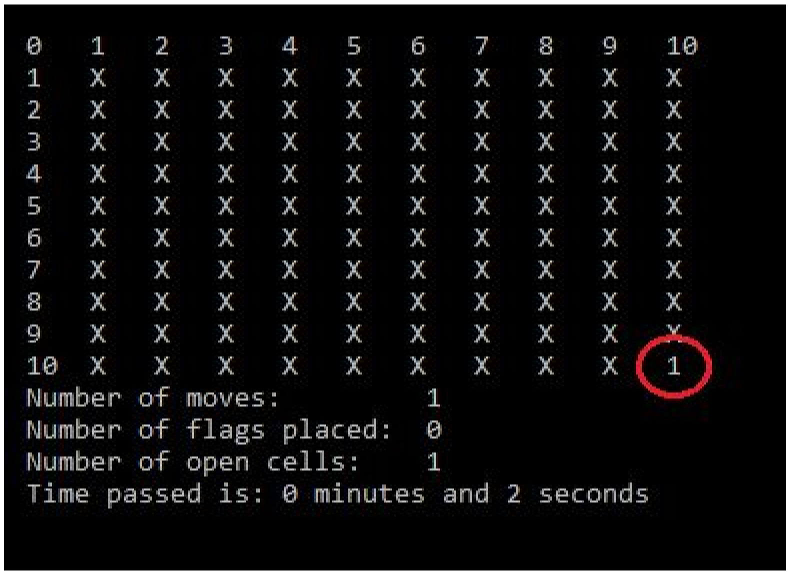
- Enter the row and column number of the desired cell to select it
- Decide what you want to do with that cell:
- Open cell
- Flag cell
- Question mark cell
- Unmark cell
- Save
- Main menu



```
0   1   2   3   4   5   6   7   8   9   10
1   X   X   X   X   X   X   X   X   X   X
2   X   X   X   X   X   X   X   X   X   X
3   X   X   X   X   X   X   X   X   X   X
4   X   X   X   X   X   X   X   X   X   X
5   X   X   X   X   X   X   X   X   X   X
6   X   X   X   X   X   X   X   X   X   X
7   X   X   X   X   X   X   X   X   X   X
8   X   X   X   X   X   X   X   X   X   X
9   X   X   X   X   X   X   X   X   X   X
10  X   X   X   X   X   X   X   X   X   X
Number of moves:         0
Number of flags placed:  0
Number of open cells:    0
Time passed is: 0 minutes and 0 seconds

 enter the desired row: 10
enter the desired column:       10
choose action:
 1- open cell
 2- flag cell
 3- question mark cell
 4- unmark cell
 5- save
 6- Main menu
=>       1
```

Figure (2-1)



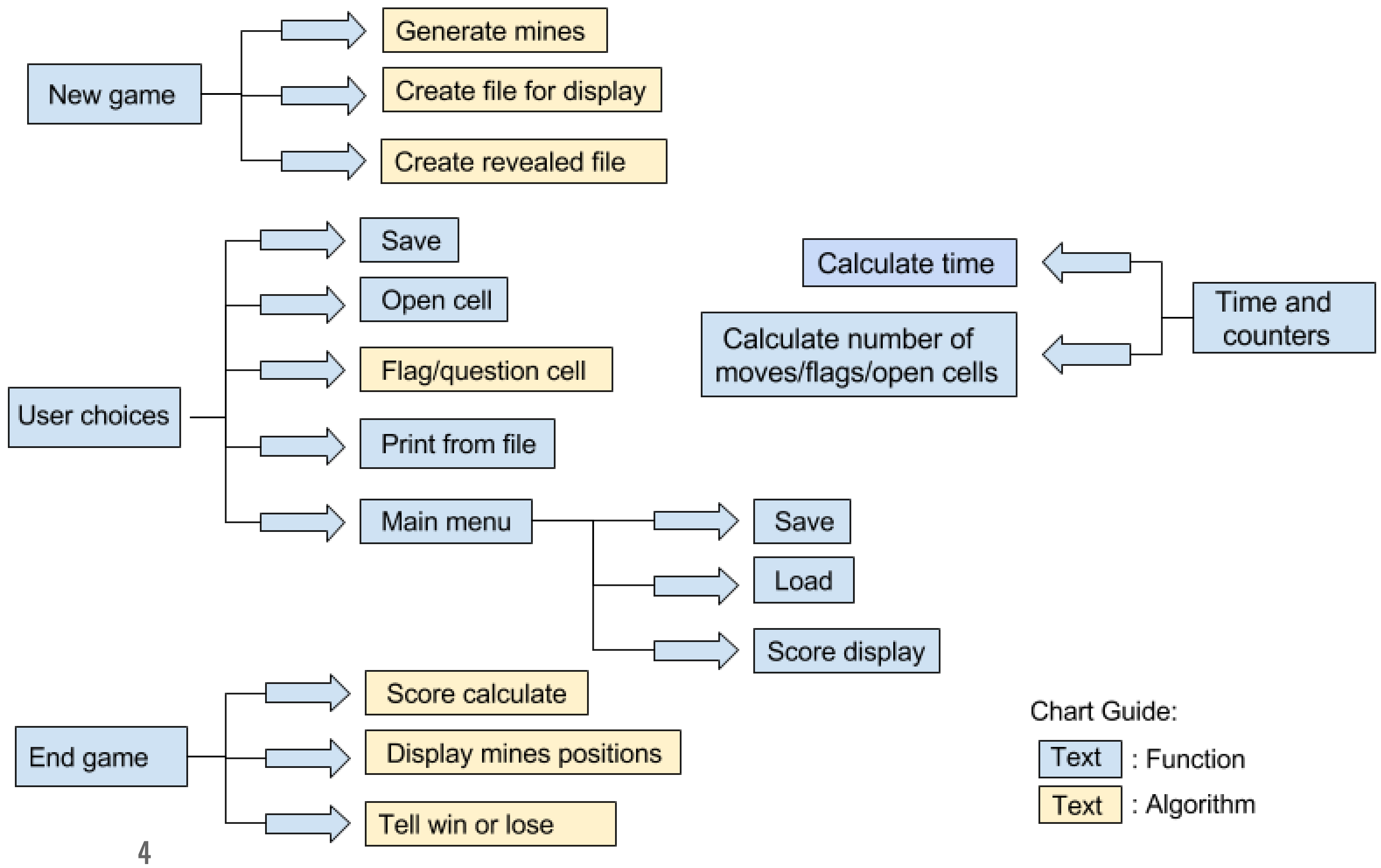
```
0   1   2   3   4   5   6   7   8   9   10
1   X   X   X   X   X   X   X   X   X   X
2   X   X   X   X   X   X   X   X   X   X
3   X   X   X   X   X   X   X   X   X   X
4   X   X   X   X   X   X   X   X   X   X
5   X   X   X   X   X   X   X   X   X   X
6   X   X   X   X   X   X   X   X   X   X
7   X   X   X   X   X   X   X   X   X   X
8   X   X   X   X   X   X   X   X   X   X
9   X   X   X   X   X   X   X   X   X   X
10  X   X   X   X   X   X   X   X   X   1
Number of moves:         1
Number of flags placed:  0
Number of open cells:    1
Time passed is: 0 minutes and 2 seconds
```

Figure (2-2)

Overview of the code:

The code was designed as a combination of functions and algorithms, to make the game user friendly together. The main functions and algorithms as shown in figure (3-1), are:

- **Start game**: takes in the grid size and randomly generates the places of the mines, saves the generated puzzle revealed in a file for further use from the game's end while creating another file with "X"s filling the puzzle, along side with an index for lines and columns.
- **User choices**: Takes a choice from user and act accordingly. Also makes it impossible for the first chosen cell to be a mine.
- **End game**: Tells lose or win and calculate score and rank.
- **Time and counter**s: calculates time since the start of the game and the number of moves/flags/open cells.
- **Print file**: Takes in the name of any file and prints it onto the screen.



Figure(3-1)

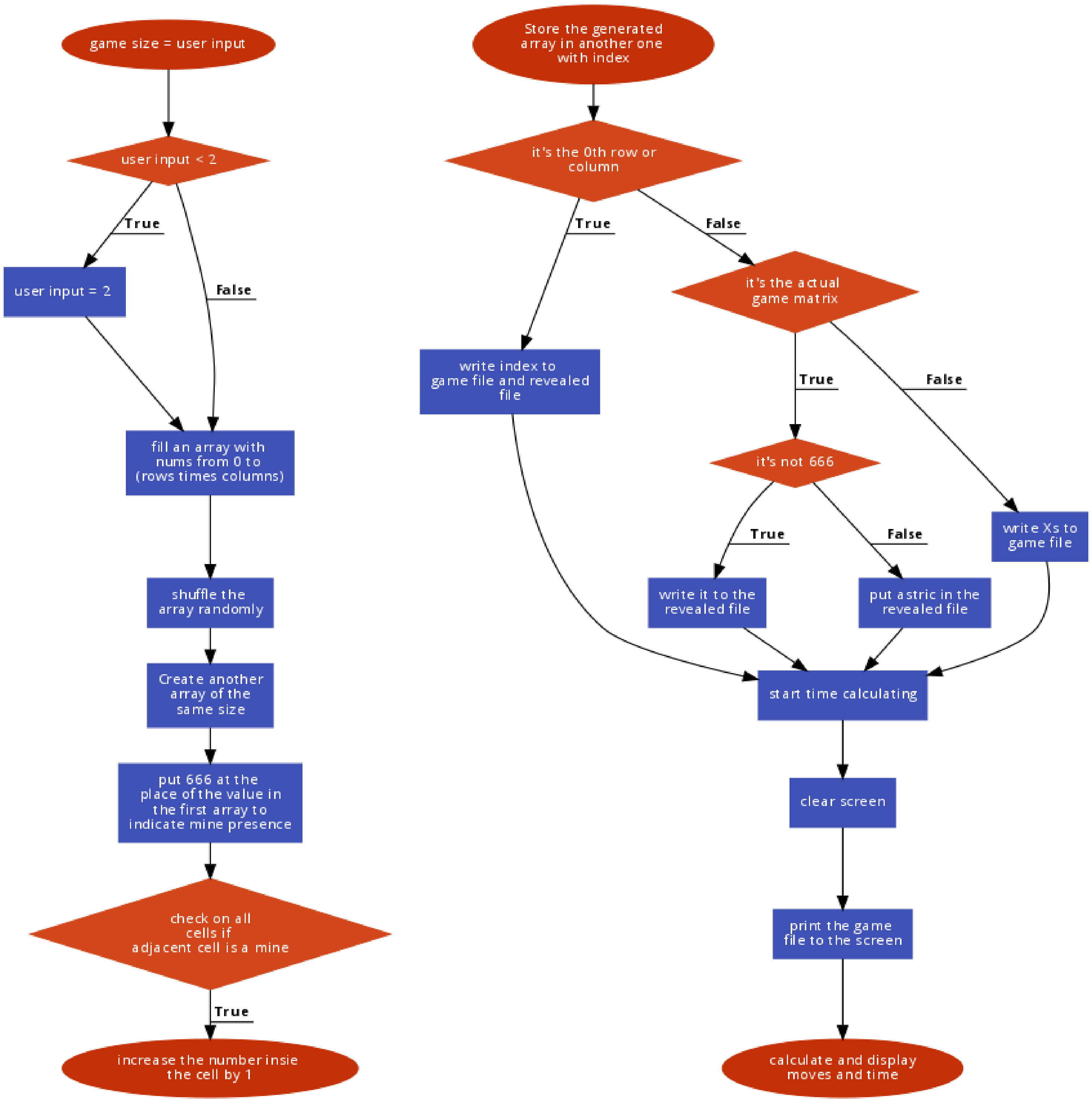
Main data structures and functions

Name	Type	Use
gamefile	.txt file	Stores what's being displayed to the user
mines_display	.txt file	Stores the actual generated puzzle (all cells opened)
flagsDisplay	.txt file	Stores number of moves/flags/open cells and time passed
save	.txt file	Stores "gamefile.txt" inside it whenever the user choses to do so for further use.
values_array	char*	Stores "mines_display.txt" as an array for compare/modify operations
game_array	char*	Stores the "gamefile.txt" as an array for compare/modify operations
opencell	Void function	Opens cells when user chooses so, or when adjacent to a 0 containing cell, or when an adjacent cell has adjacent flags equals to its numbers.
flag_check	Void function	Checks if number of adjacent flags to a cell equals its numbers, if so, it calls opencell

Name	Type	Use
endgame	Void function	Takes win or lose as 0 or 1 respectively, and acts accordingly, displaying time and number moves/flags/open cells, and displays where the mines were Then calculates the score
savefile	Void function	Copies any file into any other file, in this case, “gamefile.txt” to “save.txt”
loadfile	Void function	The same as save file, but from “gamefile.txt” to “savefile.txt”
mainmenu	Void function	Gives options of starting the game, loading, and displaying scores.
game_check	Void function	Checks if the game has ended or not, either by lose or win
display_timeANDflags	Void function	Calculates time and number moves/flags/open cells and stores them in “flagsDisplay.txt”
start_game	Void function	(Covered in page 4)
file_characters_count	Int function	Calculates the number of characters in any file and returns it Used to define arrays of its size to store files into
user_choices	Void function	(Covered in page 4)

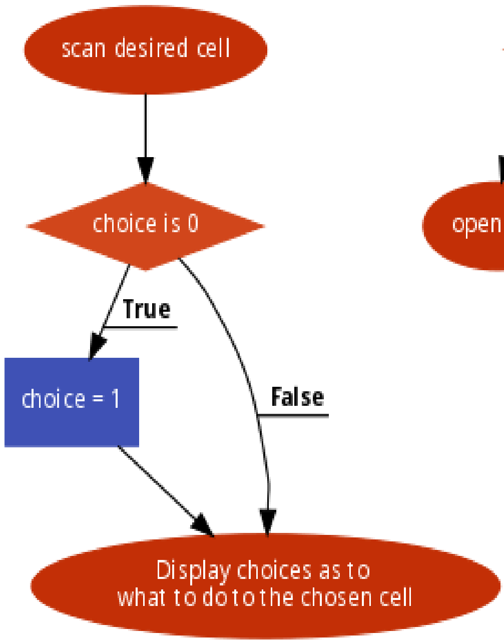
Flow charts

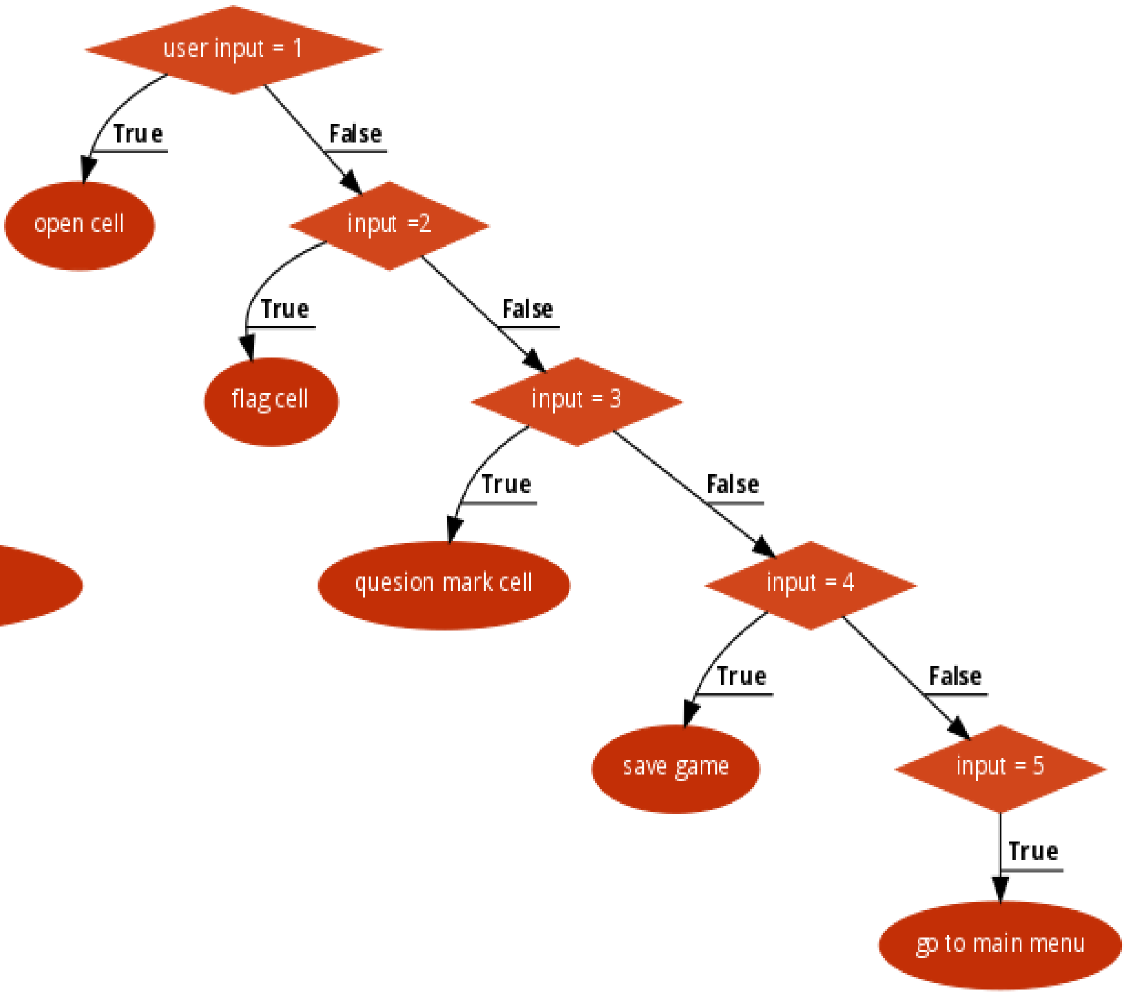
Start game:



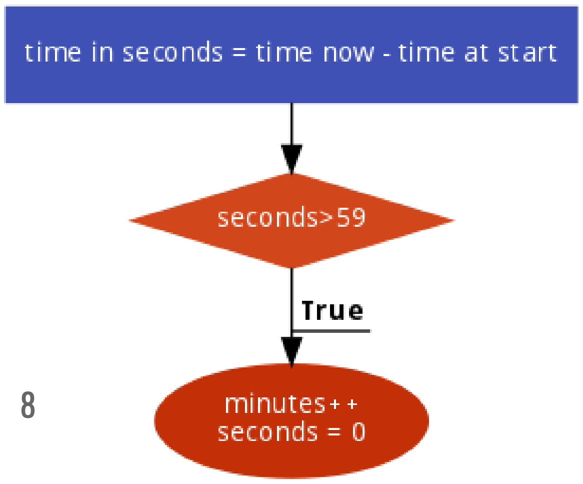
game size = user input
user input < 2
True
False
user input = 2
fill an array with nums from 0 to (rows times columns)
shuffle the array randomly
Create another array of the same size
put 666 at the place of the value in the first array to indicate mine presence
check on all cells if adjacent cell is a mine
True
increase the number insie the cell by 1
Store the generated array in another one with index
it's the 0th row or column
True
False
write index to game file and revealed file
it's the actual game matrix
True
False
it's not 666
True
False
write it to the revealed file
put astric in the revealed file
write Xs to game file
start time calculating
clear screen
print the game file to the screen
calculate and display moves and time

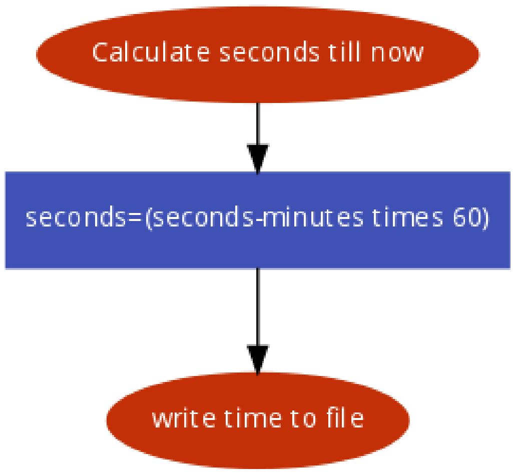
User choices:





Time:



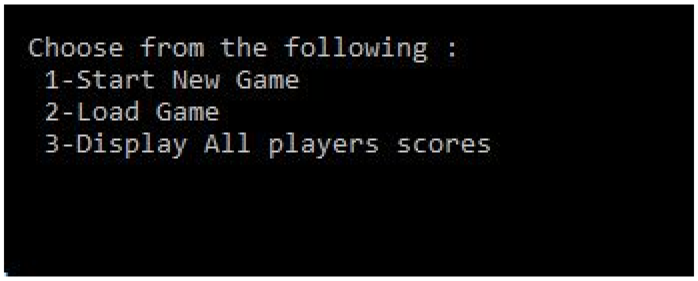

Calculate seconds till now
seconds=(seconds-minutes times 60)
write time to file

Game manual:

First:

As soon as you start the game, the main menu -shown in fig. (4-1)- will appear asking you what you want to do.

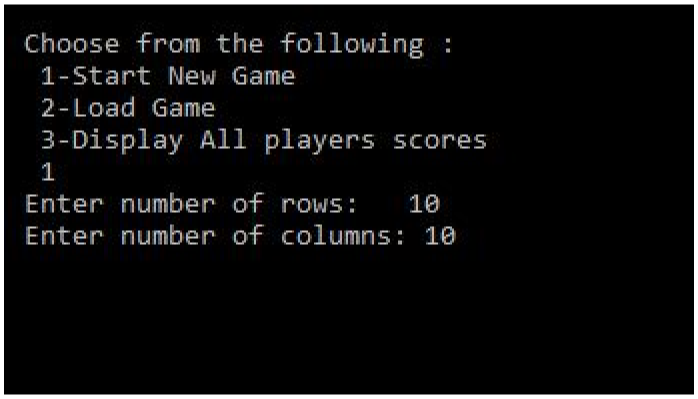
Press The number of the choice you want followed by ENTER to chose it.



```
Choose from the following :
 1-Start New Game
 2-Load Game
 3-Display All players scores
```

` Figure(4-1)

On choosing the first choice, you will be asked to choose difficulty, by choosing the game size, entering the number of rows you need followed by ENTER, then number of columns followed by ENTER as well As shown in figure (4-2)



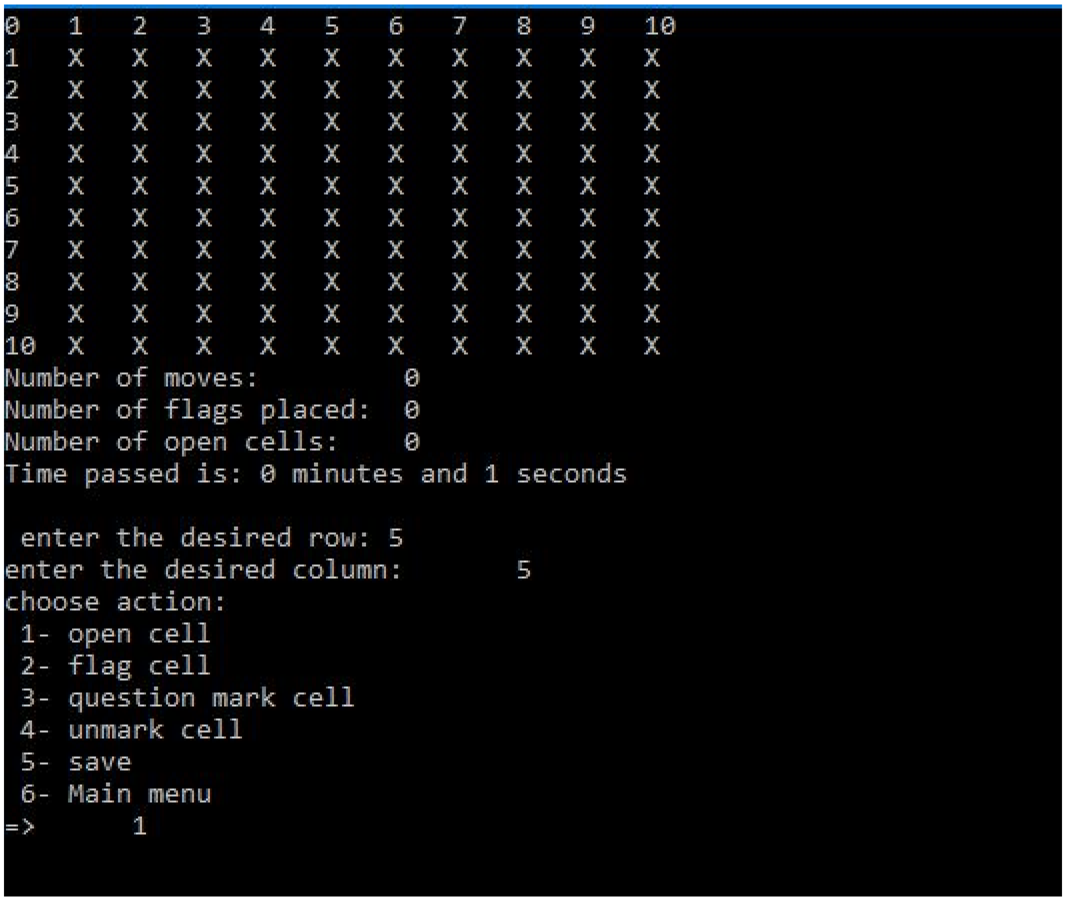
```
Choose from the following :
 1-Start New Game
 2-Load Game
 3-Display All players scores
 1
Enter number of rows:   10
Enter number of columns: 10
```

figure (4-2)

Next:

The game will generate, as shown in figure (4-3), the “X”s indicate closed cells.

Under the game grid, the time passed, number of moves, flags placed and open cells will be visible.



```
0   1   2   3   4   5   6   7   8   9   10
1   X   X   X   X   X   X   X   X   X   X
2   X   X   X   X   X   X   X   X   X   X
3   X   X   X   X   X   X   X   X   X   X
4   X   X   X   X   X   X   X   X   X   X
5   X   X   X   X   X   X   X   X   X   X
6   X   X   X   X   X   X   X   X   X   X
7   X   X   X   X   X   X   X   X   X   X
8   X   X   X   X   X   X   X   X   X   X
9   X   X   X   X   X   X   X   X   X   X
10  X   X   X   X   X   X   X   X   X   X
Number of moves:         0
Number of flags placed:  0
Number of open cells:    0
Time passed is: 0 minutes and 1 seconds

 enter the desired row: 5
enter the desired column:       5
choose action:
 1- open cell
 2- flag cell
 3- question mark cell
 4- unmark cell
 5- save
 6- Main menu
=>      1
```

Figure (4-3)

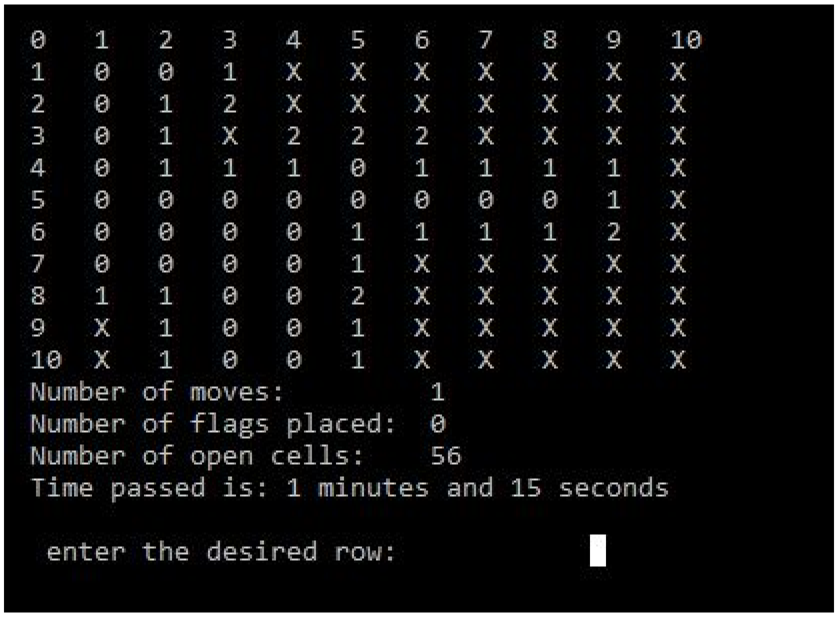
Afterwards you get to make the choices to play the game:

First you need to specify a cell by entering the number of rows then the number of columns. The index should guide you in choosing.

Then you need to tell the game the action to perform on that cell as shown in figure (4-3); by entering the number corresponding to the action followed by ENTER.

Choose any square, preferably towards the middle. Most Minesweeper players click random squares until a group of squares "opens" up. If 4 or 5 squares opened after your click, it's time to evaluate the numbers. If only one square opens after your click, find another random square.

Check the numbers to find bombs. As you uncover tiles, you will see numbers revealed. A number means that there are that number of bombs touching that tile (both sides, top/bottom, and diagonally). If you see a 1 on the board, it means that square is touching exactly 1 mine.



```
0   1   2   3   4   5   6   7   8   9   10
1   0   0   1   X   X   X   X   X   X   X
2   0   1   2   X   X   X   X   X   X   X
3   0   1   X   2   2   2   X   X   X   X
4   0   1   1   1   0   1   1   1   1   X
5   0   0   0   0   0   0   0   0   1   X
6   0   0   0   0   1   1   1   1   2   X
7   0   0   0   0   1   X   X   X   X   X
8   1   1   0   0   2   X   X   X   X   X
9   X   1   0   0   1   X   X   X   X   X
10  X   1   0   0   1   X   X   X   X   X
Number of moves:         1
Number of flags placed:  0
Number of open cells:    56
Time passed is: 1 minutes and 15 seconds

 enter the desired row:
```

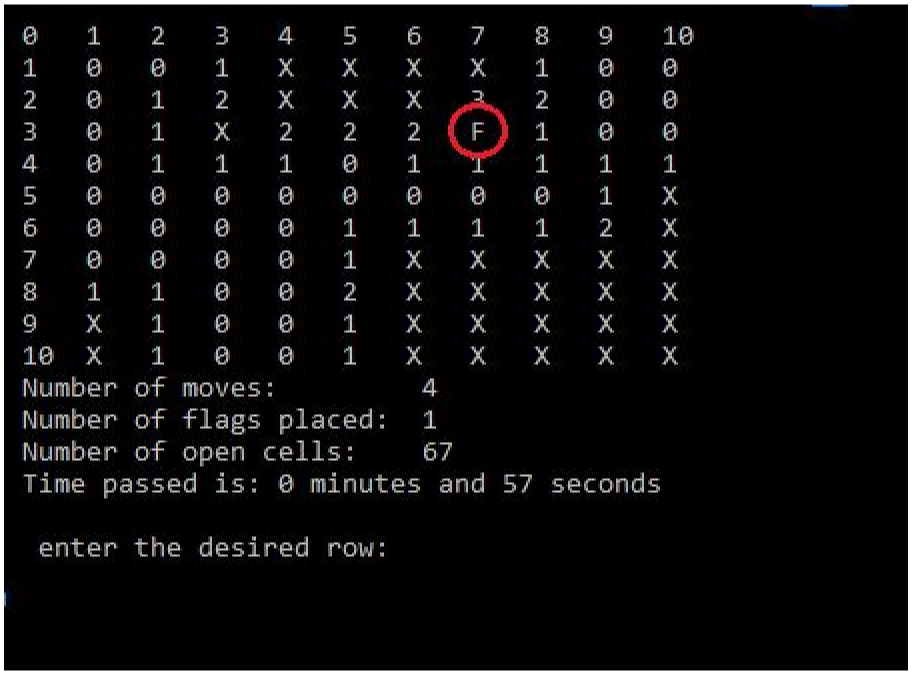
Figure (4-4)

Click known safe squares. Eliminate squares that can't possibly contain mines by opening them. Say you found a 1, and you're pretty sure where the mine for that 1 is. You can click all the other squares around that 1 to open them, because the 1 can only be in contact with a single mine.

- Use all of the numbers in a given area to figure out where the mines are.

Flag cells to identify mines. When you're reasonably certain you've found a mine, identify it by flagging it. This will put a letter "F" on the mine, So you know you shouldn't open it, as shown in the opposite figure.

Notice that, if you flag as many cells around a cell as its number, the rest of the adjacent cells to it will open; even if they contain mines.



```
0   1   2   3   4   5   6   7   8   9   10
1   0   0   1   X   X   X   X   1   0   0
2   0   1   2   X   X   X   3   2   0   0
3   0   1   X   2   2   2   F   1   0   0
4   0   1   1   1   0   1   1   1   1   1
5   0   0   0   0   0   0   0   0   1   X
6   0   0   0   0   1   1   1   1   2   X
7   0   0   0   0   1   X   X   X   X   X
8   1   1   0   0   2   X   X   X   X   X
9   X   1   0   0   1   X   X   X   X   X
10  X   1   0   0   1   X   X   X   X   X
Number of moves:         4
Number of flags placed:  1
Number of open cells:    67
Time passed is: 0 minutes and 57 seconds

 enter the desired row:
```

Keep moving through a process of elimination. As you go around the board, flag potential mines and open cells.

Win and loss:

If you open all the cells that are not containing mines, You win the game.

If you open a cell that contains a mine you lose.