Documentation for Minesweeper

## For Developers

## gameBoars, boardSize.

These variables are used in a global scope, so that no time is wasted in passing them to functions again and again. These boards holds all the values with which the user interacts and are then displayed onto the screen.

## minedBoards

These boards are used only to hold mines. The user makes no interaction with them throughout the program and most logical part is being held with the minedBoards, since the game is all about mines and the mines reside here.

## currentChoice

This variable is used in half the UX and logical functions, to handle the boards and user’s input

### startGame()

The game starts with the startGame( ) function.

Instead of starting the game from main function, a separate startGame() function has been used, so that the game can be replayed after winning or losing it.

### printMenu() ,printLines(), printinstructions(),setGameState(), printTop()

are all UI functions. They provide the Interface for the game.

printMenu() is a UI function that prints the main menu, it calls the printTop() function, printMenu() also sets the UX part by inputting the currentChoice. Thus joining the UI and UX

printTop()is used throughout the program with all the UI/UX functions so that a unique UI is maintained throughout the game.

### setGameState()

sets the game state based on the .. currentChoice variable.

### Newgame()

This function is called whenever a new game is going to start, whenever user selects to play the game. It initializes all the boards and sets mines on the board, prints the board and then moves on to the UX part where it takes the row, column and the required operation . Then moved on to the checkInput(), where it checks the

### Takeinput()

This function, first checks if the user has won the game. If user has won the game, then moves on to the

**rePlay()** function where it prompts for playing again or exiting the game.

This function uses **winCheck** var, which is reset to zero as the **startGame()** function is called.

### checkInput()

**takeInput()** calls **checkInput(),** function which checks the users input and the operation

It opens the block, flags or removes the flag, based on the operation selected.

### isValid()

This function checks wheter the row and column on which the operation is going to be performed is a valid cell or not. It is used by most of the logic based functions.

### showEmpty()

It is a recursive function, which checks for the empty 8 neighbors and thenrecursively checks the empty neighbors .

Then it finally displays the mine count on the neighbors .

It also uses countMines function to show the neighbors that are empty but have mined neighbors.

### setEmpty()

makes the value in main game board to be empty

### isEmpty()

checks whether the cell has is completely empty or not. It’s neighbors also should not have a mine.

### isFlagged()

checks the cell for flag.Used in checkInput, so the user should not open a flagged cell, without removing the flag first.

### printMines()

Mainly used for debugging, to check whether the input works as per requirement or not. Also used to cheat and check whether the Win condition works or not.

When game is over, this function is called to print all the Mines to the user.

## replay()

This function asks the user’s opinion after winning or losing the game.

User can choose to play again or quit.

### countMine()

This function counts the mines all around the cell (in the 8 neighbors) and returns the number of mines around the cell.

## Issues/bugs found

If user enters a character value instead of numeric , in row,col or operation. The program runs into an infinite loop.

In **showEmpty(),** half the code works recursively. The cells that are north, north west, north east, north west, and east direction works in recursive way.

The other direction go to an infinite loop or throws stack overflow exception and can only be dealt by using  **try catch** blocks, which are not a part of our course curriculum and thus not used here.

All these bugs can be handled only by the use of **try catch** blocks but since they aren’t a part of our course, we are limited to use the program with these bugs.

Rest the whole program works just fine and there is no issue with the logic at any place.

# For Users

This program is a representation of the old Minesweeper game by Microsoft. This program is based on console and doesn’t involve any GL, Graphics Libraries.

This program comes with a main menu, which asks for user’s input and proceeds on to the next menu.

The only known bug is that you should not enter a character value into the program at any time.

When you successfully open all the cells with no mines, and flag the mined ones, you win this game

THE END