**Final Project Report**

**Mine Sweeper**

**Overview:**

The Application is the game Minesweeper.. The game plots a 2D grid containing cells, each cells may or may not contain a mine, the player must open all cells without mines to win, if the player opens a cell with a main, he loses.

Upon opening the game offers the player 3 options, start a new game, load a saved game. Or display the score Board. Plus the option of exiting the application.

If the user chooses a new game the application asks him for his username and the size of the grid he'd like to play, the grid should be at least 3x3 and at most 1000x1000 in size. If the player wins, his score is calculated and he is added to the scoreboard and the final Ranks are displayed. If the player lost he is not added and is given the option to again start a new game (with a new username if he chooses).

During the Game:

The player has 6 options, Open a cell, Flag a cell, Question a cell, Unmark a cell, Save the game or exit without saving.

Open cell:

The player can open a cell only if it is not flagged, upon opening there 3 possibilities. It had a mine, then the player loses. It has mines in neighboring cells, then it opens to the number of neighboring mines, or last it was empty, then all cells around it are open till ones with neighbouring mines.

**Flag Cell:** Player should opt to that when he believes a cell has a mine, then he is not allowed to open it later and a letter 'F' is displayed.

**Question Cell:** Player should opt to that when he believes a cell 'might' have a mine, he, however, is still allowed to open it later and a '?' is displayed.

**Unmark:** Player can remove the flag or question on a cell.

**Save:** Player can save game to return and open it later.

**Exit w/o Save:** Player can quit a game without saving.

**Data Structures:** Basic data types in C were used from structures, strings, structure arrays, string arrays, etc..

**Functions:**

**void loadSavedGames:**

uploads Names of files of saved games from text file (savedgames.txt) to an array of strings in main interface.

**void saveGame:**

Takes game info (The grid Data, Start time and player username) and savesit to a binary file.

**Void LoadGame:**

Loads game info previously saved and assigns them to pointers of respective variable in main.

**void sortScores:**

Sorts players according to their scores using a bubble sort algorithm.

**void saveScores:**

Saves sorted scores to text file (scores.txt)

**// Note:** sortScores & saveScores are only called should a player win a game. Guaranteeing that scores are only saved to the text file sorted and loaded while sorted.

**void scoreboard:**

Called at the start of program (like loadSavedGames) and reads scores from text file into an array of structs.

**void printScores:** Prints the array of players with ranks.

**Int takenName: *int takenName(player\* list, int numPlayers, char\* input)***

called when new player enters username to check if username was already taken. It returns 1 if there is a same name and 0 if not.

**Pseudo code:**

Set i=0

While i is smaller than number of players

If input string equal to the i'th string

Return true

Return false

**Int isInt: *int isInt(char\* test)***

At start of a new game, when user enters grid dimensions, function validates they are indeed numbers within supposed bounds, if user enters a nondigit, or if not within bound, it returns zero, otherwise it returns the string he entered as a number.

**Pseudocode:**

Set i=0

While I is smaller than or equal to string length

If string is 1 character length and character is less than 3

Return false

If i'th character isn't between 0 & 9

Return false

Return Number form of string.

**Int parseToInt:**

Like isInt only it doesn't return false if number was less that 3, this is for grid dimension during playing.

**Pseudocode:**

Set i=0

While I is smaller than or equal to string length

If i'th character isn't between 0 & 9

Return false

Return Number form of string.

**Float calcScore:**

Returns player score using formula if player has won.

**Void initGrid:**

Sets grid cells to default values at start of new game.

**Int fillMines:**

Randomly fills mines at start of new game.

**Void incrementcontent:**

Assigns number in cells to number of adjacent mines.

**PseudoCode**:

Set i=0 k=0

While i less than or equal to number of rows

While k less than or equal to number of columns

If cell at I has mine

Add 1 to all cells next to it

**Void lose:**

**Pseudocode:**

Set i=0 k=0

Print(You lose)

While i less than or equal to number of rows

While k less than or equal to number of columns

If cell at I has mine

Set 'display' to '\*'

If cell has mine and was opened

Set 'display' to '!'

If cell had mine and was flagged

Set display to '-'

Else set 'display' to 'M'

Display grid

**Void win:**

**Pseudocode:**

Set i=0 k=0

Print (You win)

While i less than or equal to number of rows

While k less than or equal to number of columns

If cell at I has mine

Set 'display' to 'F'

Else set 'display' to '-'

Display grid

**Void FlagCell & void question cell:**

Change display of cell to 'F' & '?' respectively, also change its status to flagged or questioned.

**Void isFlagged: *int isFlagged(cell\*\* grid, int row, int col, int x, int y)***

Checks user input for flagging a cell, it returns 0 if input was out of bound or if cell is opened.

**Pseudocode:**

If x bigger than row or smaller than 1 or y bigger than col or smaller than 1

Return false

If cell is opened return false

If cell is flagged

Print (already flagged) & return false

Return true

**Void isQuestioned: *int isQuestioned(cell\*\* grid, int row, int col, int x, int y)***

Checks user input for flagging a cell, it returns 0 if input was out of bound or if cell is opened.

**Pseudocode:**

If x bigger than row or smaller than 1 or y bigger than col or smaller than 1

Return false

If cell is opened return false

If cell is questioned

Print (already questioned) & return false

Return true

**Void unmark:**Removes flag or question on cell and returns display to 'X'

**Void isMarked:** Checks if input by user is flagged or questioned. Returns 0 if cell is neither flagged or questioned and prints error message.

**Void displayGrid:**

**Int validateInput:**

Checks user input for opening a cell, it returns 0 if input was out of bound or if cell was already opened or if it were flagged.

**Pseudocode:**

If x bigger than row or smaller than 1 or y bigger than column or smaller than 1

Return false

If cell is opened print(already open) & return false

If cell is flagged

Print (flagged cant open) & return false

Return true

**Void openCell:**

***void openCell(cell\*\* grid, int row, int col, int x, int y, int\* openCells, int\* ques)***

Opens cell. If it had a mine, then the player loses. It has mines in neighboring cells, then it opens to the number of neighboring mines, or last it was empty, then all cells around it are open till ones with neighboring mines.

**Pseudocode:**

If x bigger than row or smaller than 1 or y bigger than column or smaller than 1

Return

If cell is opened

Return

If cell is Flagged

Return

Set 'display' to content of cell

Mark cell as open

Increment openCells

if cell had no mines around

set display to " "

open all adjacent cells recursively.

**Main Function:**

**Pseudocode:**