/\* Title: Aggravating Squares

Written By: Sam Graham

Description: This is a non-gui game that lets the user pick the size of thier 'board' (a graph).

They must then navigate through all of the squares without traveling through a

square they previously went through. \*/

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

int inputBoxesPerRow;

int inputNumOfRows;

int inputStartRow;

int inputStartBox;

int currentRow;

int currentBox;

char box[2] = { '\_', '|' };

char possiblePlayerLocation[1000][1000] = {'a'};

char markedBox = '\*';

/\* This function builds the player's board. It uses input to determine the size of the board and an reads indexes of the 2 dimensional player location array to determine where to display an asterick '\*' signaling to the player that they have already been to that square.\*/

int display\_the\_board(void)

{

printf("\n\t ");

for (int columnCount = 0; columnCount < inputBoxesPerRow; columnCount++)

{

printf("%d ", columnCount + 1);

}

printf("\n");

for (int numOfRows = 0; numOfRows < inputNumOfRows; numOfRows++)

{

printf("\t%2d |", numOfRows + 1);

for (int boxesPerRow = 0; boxesPerRow < (inputBoxesPerRow \* 2); boxesPerRow++)

{

if (boxesPerRow % 2 == 0)

{

if (possiblePlayerLocation[numOfRows][boxesPerRow] == markedBox)

printf("%c", markedBox);

else

printf("%c", box[0]);

}

else

printf("%c", box[1]);

}

printf("\n");

}

printf("\n");

return 0;

}

/\* This method checks all the possible squares to see if the player has made it through them all.\*/

int check\_for\_winner(void)

{

int squareIndex = 0;

int completedSquares = 0;

for (int rowIndex = 0; rowIndex < inputNumOfRows; rowIndex++)

{

for (int squares = 1; squares < inputBoxesPerRow + 1; squares++) /\*squares is initialized to 1 to offset the algorithm in the if else, inputBoxesPerRow + 1 to accommodate\*/

{

if (squares % 2 == 0)

{

squareIndex = (squares \* 2) - 2;

}

else

{

squareIndex = (squares -1 ) \* 2;

}

if (possiblePlayerLocation[rowIndex][squareIndex] == markedBox)

{

completedSquares++;

}

}

}

printf("%d Squares completed", completedSquares);

if (completedSquares == inputNumOfRows \* inputBoxesPerRow)

printf("\n\n Congratulations, YOU ARE A WINNER!!");

return 0;

}

/\* This function displays some general rules for the game. It asks for user input that is fed to the display board function to build the players game. It asks for the user to choose a starting point and gives some feedback if they choose something off the board. Once the player has a starting point, it reads controls from the arrow keys to let the user navigate the board. \*/

int main()

{

printf("\nWelcome to Aggravating Squares.\nWritten By: Sam Graham, Tampa, FL USA\n\nIn this game you pick the size your gameboard (hint: its a two dimensional graph).\n");

printf("A template of your board will display. You then choose a starting location. \nTry make your way through every square without crossing through a square you've already traversed.\n");

printf("Another hint: some start positions may make this an impossible task\n\n");

printf("How many boxes would you like in each row? ");

scanf("%d", &inputBoxesPerRow);

printf("How many rows would you like? ");

scanf("%d", &inputNumOfRows);

display\_the\_board();

printf("\nWhat row would you like to start on? ");

scanf("%d", &inputStartRow);

while (inputStartRow <= 0 || inputStartRow > inputNumOfRows) //error checking for valid numeric input

{

printf("Please choose a row that is on the board! \n Give me another number: ");

scanf("%d", &inputStartRow);

}

printf("Which box in that row would you to begin in? ");

scanf("%d", &inputStartBox);

while (inputStartBox <= 0 || inputStartBox > inputBoxesPerRow) //error checking again

{

printf("Please choose a box that is on the board! \n Give me another number: ");

scanf("%d", &inputStartBox);

}

currentRow = inputStartRow - 1;

if (inputStartBox % 2 != 0)

currentBox = (inputStartBox -1) \* 2;

else

currentBox = (inputStartBox \* 2) - 2;

possiblePlayerLocation[currentRow][currentBox] = markedBox;

display\_the\_board();

printf("Use the arrow keys on your keyboard.");

int ch;

/\* This loop reads the users input from the arrow keys. It makes sure the user stays on the board, and that they are not going through the same square twice. A new board is put to the screen after every move. \*/

while ((ch = getch()) != '#')

{

switch (ch)

{

case 72:

{ printf("Press '#' to exit");

currentRow--;

if (currentRow < 0)

{

printf("\nTry to keep it on the board\a");

currentRow++;

}

else

{

if (possiblePlayerLocation[currentRow][currentBox] == markedBox)

{

printf("\nYou have already been to that square. You cannot go through it again.");

currentRow++;

}

else

{

possiblePlayerLocation[currentRow][currentBox] = markedBox;

}

}

display\_the\_board();

break;

}

case 80:

{

printf("Press '#' to exit");

currentRow++;

if (currentRow > inputNumOfRows - 1)

{

printf("\nTry to keep it on the board\a");

currentRow--;

}

else

{

if (possiblePlayerLocation[currentRow][currentBox] == markedBox)

{

printf("\nYou have already been to that square. You cannot go through it again.\a");

currentRow--;

}

else

{

possiblePlayerLocation[currentRow][currentBox] = markedBox;

}

}

display\_the\_board();

break;

}

case 75:

{

printf("Press '#' to exit");

currentBox-=2;

if (currentBox < -1)

{

printf("\nTry to keep it on the board\a");

currentBox += 2;

}

else

{

if (possiblePlayerLocation[currentRow][currentBox] == markedBox)

{

printf("\nYou have already been to that square. You cannot go through it again.\a");

currentBox += 2;

}

else

{

possiblePlayerLocation[currentRow][currentBox] = markedBox;

}

}

display\_the\_board();

break;

}

case 77:

{

printf("Press '#' to exit");

currentBox+=2;

if (currentBox >= inputBoxesPerRow \* 2)

{

printf("\nTry to keep it on the board\a");

currentBox-= 2;

}

else

{

if (possiblePlayerLocation[currentRow][currentBox] == markedBox)

{

printf("\nYou have already been to that square. You cannot go through it again.\a");

currentBox -= 2;

}

else

{

possiblePlayerLocation[currentRow][currentBox] = markedBox;

}

}

display\_the\_board();

break;

}

}

}

return check\_for\_winner();

}