Assignment 1

Pedram Pasandide

Due Date: 16 October 2023

In this assignment, you are tasked with implementing the classic game of Minesweeper in C. Minesweeper is a classic computer game that comes pre-installed on many Windows operating systems. The objective of the game is to clear a grid of cells without detonating hidden mines. Players click on cells to reveal numbers, which indicate how many mines are adjacent to that cell. By using logic and deduction, players must flag cells they suspect contain mines and uncover safe cells until they've cleared the entire grid or flagged all the mines.

Minesweeper is a popular puzzle game that challenges players to think strategically and logically while avoiding mines. It's a great example of a simple yet engaging game that can be implemented as a programming project, especially for learning purposes. Students can create their own text-based version of Minesweeper in C, which would involve working with 2D arrays, loops, and conditional statements to manage the game grid and implement game rules.

A. (2 points) Write all your codes in a single file named `main.c`. In this assignment you need to implement a simplified version of the Minesweeper game in C using a 10x10 array. Declare a 2D array variable with `char board[SIZE][SIZE]`, where `#define SIZE 10` is defined at the top of you code. You can initialize all the element of `board` array with `'-'`. The code must generate hidden bombs randomly, using `srand()` function. You can define a function called `void placeBombs(char board[SIZE][SIZE])`, to receive the `board` array an place 15 bombs (`#define BOMBS 15`) **randomly** in this array and replacing `'-'`s with `'X'`s.

B. (6 points) Then, the game must allow the player to input coordinates to reveal cells one by one. In this part the code must be able to handle potential unacceptable inputs. The `board` array is a 10by10 matrix. It means the location of element must be an index between 0 to 9 (indexing in C starts with zero!). If the input is out of this range it must print a proper message and `continue` to the next iteration. The other error might be when the input is the location of an element that it is already revealed! Then, if `board[i][j] == 'X'`, where `i` and `j` are index inputs given by user, it means you hit a bomb and the game is over! If not, in every iteration, the code must reveal the number of bombs adjacent to a revealed cell. Every element with (i,j) index in array `board` is surrounded by 8 elements:

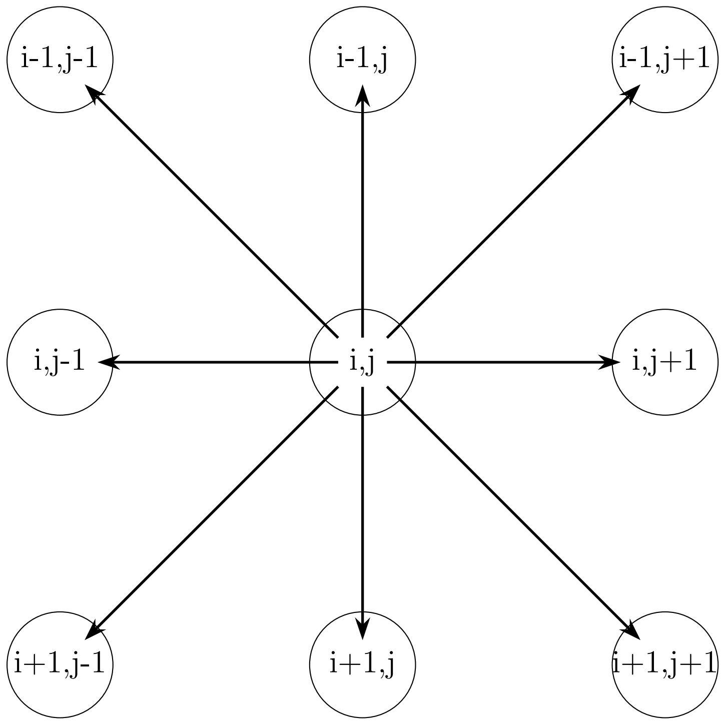

Figure 1: An element inside array `board`.

Where i represents the rows and j is showing the columns in `board` array. So, the elements around board[i,j] are:

- board[i-1,j]
- board[i+1,j]
- board[i,j-1]
- board[i,j+1]
- board[i+1,j+1]
- board[i-1,j-1]
- board[i-1,j+1]
- board[i+1,j-1]

If the user chose (i,j) and it is not a bomb, then if there is/are `n` bomb(s) in the above mention elements, it should print the number `n` in terminal.

To do this part you can define a function called:

```
int countAdjacentBombs(char board[SIZE][SIZE], int x, int y)
{
 // write the code here and call this function in every
    iteration that user makes a new guess
```

```
  }
```

C. (2 points) The game continues until the player wins or hits a bomb. Then, print out the updated `board` with locations of all bombs after the game is over (whether the user wins or loses).

Here is an example how you code must look like in terminal. First, I start the game and I see the following message:

```
Welcome to Minesweeper!
Enter the coordinates (x y) to reveal a cell.
  0 1 2 3 4 5 6 7 8 9
0 - - - - - - - - - -
1 - - - - - - - - - -
2 - - - - - - - - - -
3 - - - - - - - - - -
4 - - - - - - - - - -
5 - - - - - - - - - -
6 - - - - - - - - - -
7 - - - - - - - - - -
8 - - - - - - - - - -
9 - - - - - - - - - -
Enter coordinates (x y):
```

Then, the program is asking user to give the input at the line:

```
Enter coordinates (x y):
```

At this point any element I pick, there is a $\frac{numberofbombs}{size \times size} \times 100\%$ chance to hit a bomb. Let's say I pick the index (i,j) = (0,0). Then, this is the updated output in my terminal:

```
Welcome to Minesweeper!
Enter the coordinates (x y) to reveal a cell.
  0 1 2 3 4 5 6 7 8 9
0 - - - - - - - - - -
1 - - - - - - - - - -
2 - - - - - - - - - -
3 - - - - - - - - - -
4 - - - - - - - - - -
5 - - - - - - - - - -
6 - - - - - - - - - -
7 - - - - - - - - - -
```

```
8 - - - - - - - - - - -
9 - - - - - - - - - - -
Enter coordinates (x y): 0 0
  0 1 2 3 4 5 6 7 8 9
0 1 - - - - - - - - - -
1 - - - - - - - - - - -
2 - - - - - - - - - - -
3 - - - - - - - - - - -
4 - - - - - - - - - - -
5 - - - - - - - - - - -
6 - - - - - - - - - - -
7 - - - - - - - - - - -
8 - - - - - - - - - - -
9 - - - - - - - - - - -
Enter coordinates (x y):
```

You can see that the element I picked now is replaced by a character `'1'`. This means, in the locations (x y): 0 1, (x y): 1 0, and (x y): 1 1, there is exactly **one** bomb. Again, in the last line, the user is asked to input a new guess and I chose (i,j) = (1,1):

```
// previous part!! I didn't mention them here to avoid redundency!
Enter coordinates (x y):
Enter coordinates (x y): 1 1
  0 1 2 3 4 5 6 7 8 9
0 1 - - - - - - - - - -
1 - 2 - - - - - - - - -
2 - - - - - - - - - - -
3 - - - - - - - - - - -
4 - - - - - - - - - - -
5 - - - - - - - - - - -
6 - - - - - - - - - - -
7 - - - - - - - - - - -
8 - - - - - - - - - - -
9 - - - - - - - - - - -
```

After many iterations, this is what I see in the terminal:

```
// previous part!! I didn't mention them here to avoid redundency!
Enter coordinates (x y): 8 7
  0 1 2 3 4 5 6 7 8 9
0 1 - - - - - - - - - -
```

```
1 - 2 2 3 2 2 - - - -
2 1 1 0 0 0 1 - - - -
3 0 0 1 1 1 1 - - - -
4 0 0 1 - 1 - - - - -
5 1 1 2 - - 2 1 2 - -
6 - - - - - 1 0 1 - -
7 - - - - - 1 0 0 - -
8 - - - - - 1 2 2 1 0
9 - - - - - - - - 1 0
Enter coordinates (x y): 7 7
Invalid input. Try again.
  0 1 2 3 4 5 6 7 8 9
0 1 - - - - - - - - -
1 - 2 2 3 2 2 - - - -
2 1 1 0 0 0 1 - - - -
3 0 0 1 1 1 1 - - - -
4 0 0 1 - 1 - - - - -
5 1 1 2 - - 2 1 2 - -
6 - - - - - 1 0 1 - -
7 - - - - - 1 0 0 - -
8 - - - - - 1 2 2 1 0
9 - - - - - - - - 1 0
Enter coordinates (x y):
```

I have guessed `Enter coordinates (x y): 7 7`, which was already revealed. That's why I printed out `Invalid input. Try again.`. At some point I chose the element (i,j)=(7 0) which was a bomb. I print out `Game Over! You hit a bomb.` and the latest `board` array with the location of bombs revealed with the character `'X'`:

```
// previous part!! I didn't mention them here to avoid redundency!
Enter coordinates (x y): 7 0
Game Over! You hit a bomb.
  0 1 2 3 4 5 6 7 8 9
0 1 - X X X - X - - 0
1 X 2 2 3 2 2 - - - -
2 1 1 0 0 0 1 X - - -
3 0 0 1 1 1 1 - - - -
4 0 0 1 X 1 - X - - -
5 1 1 2 - - 2 1 2 X -
```

```
6 X - - X X 1 0 1 1 1
7 X - - - - 1 0 0 0 0
8 - - - - - 1 2 2 1 0
9 0 - - - - - X X 1 0
```

This is the general format that your code must look like:

```c
// <including necessary library(s)>
#define SIZE 10
#define BOMBS 15

// <your code if necessary>
// define the functions here for example:
int countAdjacentBombs(char board[SIZE][SIZE], int x, int y)
{
 // <your code>
}

int main()
{
 char board[SIZE][SIZE];
 int gameOver = 0; \\ a boolean checking if the game is over
 int remainingCells = SIZE * SIZE - BOMBS;
 // define here other variables you may need
 // <your code if necessary>
 while (remainingCells > 0 && !gameOver)
 {
  // <your code to get user input long as the game is on!>
 }
 // <your code if necessary>
}
```

D. (+1 Bonus) Skip this part if you are busy. This part might take a lot of your time. There are two things you can do if you wanna go beyond the course objectives. You can do either **Option 1** or **Option 2** to receive the bonus (you don't need to do the both!).

Option 1 **Automation**: You could see in the example I gave you, I was not choosing the elements randomly. If you want to diffuse a bomb you won't cut the wires randomly! The chance to hit a bomb in the first guess is $\frac{number of bombs}{size \times size} \times 100\%$. But let's say when I chose `Enter coordinates (x y): 2 2`, the number revealed was `'0'`. It means there is a zero chance to hit a bomb if I pick any elements surrendering(x y): 2 2 including (i j)= (1 1), (3 3), (1 2), (2 1), (1 3), (3 1), (2 3), (3 1). To do this part, you can keep the main code and add a function called `Autosolve(char board[SIZE][SIZE])` to solve the problem. Submit the new code with `main_AutoSolve.c`.

Option 2 Graphical Interface: Creating a graphical Minesweeper game in C involves several steps, and it's a more complex project than a simple example. You need to provide a graphical interface using the SDL library or even other libraries if you think it is better than SDL. Please note that this is a high-level overview, and you'll need to dive into SDL documentation and C programming for the specific implementation details.

Creating a full Minesweeper game with a graphical interface is a significant project that requires a good understanding of C programming, data structures, and game development concepts. SDL provides a flexible framework for handling graphics and input, making it a suitable choice for building such games. You may also want to consider using additional libraries or frameworks for specific features like audio playback and handling fonts. You can submit your code with `main_Graphic.c` and a good documentation how to compile and run you code.

Submit On Avenue to Learn following files:

- `main.c` including all the codes. You must submit this even if you do the bonus part.
- A short description (**a pdf file**) saying how the code must be compiled and executed to get the result.
- **Only** if you did bonus part, submit `main_AutoSolve.c` or `main_Graphic.c` and mention how to compile and run the code you description.

Looking for a Perfect Score?!

A perfect submission must:

- see the potential mistakes made by user and print out a message mentioning why the input is not acceptable. So your code must be able to handle errors instead of crashing or making mistakes in calculations.
- submit all the requested files! Don't be late, Do NOT Copy!