```
Script started on 2023-11-21 22:54:33+00:00 [TERM="xterm-256color" TTY="/dev/pts/0" COLUM
NS="68" LINES="85"]
\[\033[01;34m\]\w\[\033[00m\]$ pwd
/home/runner/Project-6-Minesweeper-The-Gameplay-Loop-kcp3s
\[\033[01;34m\]\w\[\033[00m\]$ ls -la
total 2544
drwxr-xr-x 1 runner runner     296 Nov 21 22:54 .
drwxrwxrwx 1 runner runner     142 Nov 21 21:54 ..
-rwxr-xr-x 1 runner runner   22368 Nov 21 22:50 a.out
-rw-r--r-- 1 runner runner    4284 Nov 21 22:49 Board.cpp
-rw-r--r-- 1 runner runner     775 Nov 21 22:44 Board.h
-rw-r--r-- 1 runner runner      17 Oct 27 20:51 .breakpoints
drwxr-xr-x 1 runner runner      12 Jan 24  2022 .cache
drwxr-x--- 1 runner runner     534 Nov 12 21:43 .ccls-cache
drwxr-xr-x 1 runner runner      68 Nov 21 20:44 .lesson
-rwxr-xr-x 1 runner runner 1252584 Nov 13 02:45 main
-rw-r--r-- 1 runner runner     959 Nov 21 20:46 main.cpp
-rwxr-xr-x 1 runner runner 1255712 Oct 27 20:53 main-debug
-rw-r--r-- 1 runner runner     449 Oct 27 20:53 Makefile
-rwxr-xr-x 1 runner runner   22368 Nov 21 18:53 minesweeper
-rw-r--r-- 1 runner runner       0 Nov 21 22:54 Patel_Project_6.log
-rw-r--r-- 1 runner runner    1426 Dec 21  2022 .replit
-rw-r--r-- 1 runner runner      81 Jan 18  2022 replit.nix
-rw-r--r-- 1 runner runner     524 Nov 18 02:02 Tile.cpp
-rw-r--r-- 1 runner runner     349 Nov 18 03:11 Tile.h
\[\033[01;34m\]\w\[\033[00m\]$ cat -n main.cpp
     1  // This should be the gameplay loop
     2  #include <iostream>
     3  #include "Board.h"
     4  int main() {
     5      srand(time(0));
     6  
     7  
     8      int width, height, minecount;
     9      std::cout << "Enter the difficulty you want to play((width, height, minecount
): ";
    10      std::cin >> width >> height >> minecount;
    11      Board playarea(height, width, minecount);
    12   do{
    13          playarea.display_unrevealed();
    14  
    15          int row, col;
    16          std::cout << "Enter a row and column to dig: ";
    17          std::cin >> row >> col;
    18  std::cout << std::endl;
    19          playarea.reveal(row, col);
    20      }   while(!playarea.gethas_won() && !playarea.gethas_lost());
    21  
    22      if(playarea.gethas_won()){
    23          std::cout << "YOU've WON!!!" << std::endl;
    24          std::cout << "Final Board: " << std::endl;
    25          playarea.display_unrevealed();
    26      }
    27      if(playarea.gethas_lost()){
    28          std::cout << "YOU've LOST!!!!!!!" << std::endl;
    29          std::cout << "Final Board: " << std::endl;
    30          playarea.print();
    31      }
    32  
    33  
    34  
    35  }\[\033[01;34m\]\w\[\033[00m\]$ cat -n Board.h
     1  // The Board.h header file
     2  // header file for the Board class
     3  #ifndef BOARD_H
     4  #define BOARD_H
     5  #include "Tile.h"
     6  
     7  class Board {
     8  private:
```

```
  int m_board_width;
  int m_board_height;
  int m_size;
  bool has_won = false;
  bool has_lost = false;
  int revealed_count;
  Tile *tiles{};
int minecount;


  void place_mines(int mine_count);
  void update_counts();

public:
  Board(); // Default to 8x8 and 10 mines and run from second constructor
  Board(int rows, int columns, int mine_count); // Custom usernum board
  ˜Board(); // When destroying delete the dynamic location
  void print() const;

  void reveal(int row, int col);
  bool gethas_won() const{
      return has_won;
  }
  bool gethas_lost() const{
      return has_lost;
  }
  void display_unrevealed();
};

#endif\[\033[01;34m\]\w\[\033[00m\]$ cat -n Board.cpp
// The implementation of the Board.h file
// implementation file for the Board class
#include "Board.h"
#include <cstdlib>
#include <ctime>
#include <iostream>

// Default constructor
Board::Board() {
  m_board_width = 8;
  m_board_height = 8;
  m_size = (m_board_width * m_board_height);
  tiles = new Tile[m_size];
  place_mines(10);
  update_counts();
}

// Destructor to destroy/delete the location at the end
Board::˜Board() { delete[] tiles; }

// User given dimentions and mine construction
Board::Board(int rows, int columns, int mine_count) {
  m_board_width = rows;
  m_board_height = columns;
  m_size = (m_board_width * m_board_height);
  tiles = new Tile[m_size];
    minecount = mine_count;
  place_mines(mine_count);
  update_counts();

  revealed_count = 0;
}

// Placing the mines both default and user defined
void Board::place_mines(int mine_count) {
  // srand(time(0));
  int random;
  int initial_mines = 0;

  while (initial_mines < mine_count) {
    random = rand() % m_size;
    if (tiles[random].get_value() != 9) {
      tiles[random].set_value(9);
      initial_mines++;
    }
  }
}

// Updating the counter
void Board::update_counts() {
  for (int i = 0; i < m_size; i++) {
    if (tiles[i].get_value() != 9) {
      int counter = 0;
      int i_row = i / m_board_width;
      int i_col = i % m_board_width;
      for (int r = -1; r <= 1; r++) {
        for (int c = -1; c <= 1; c++) {
          int rows = i_row + r;
          int cols = i_col + c;

          if (rows >= 0 && rows < m_board_height && cols >= 0 &&
              cols < m_board_width) {
            int index = m_board_width * rows + cols;
            if (tiles[index].get_value() == 9) {
              counter++;
            }
          }
        }
      }
      tiles[i].set_value(counter);
    }
  }
}
// Printing the board
void Board::print() const {
  std::cout << " ";
  for (int i = 0; i < m_board_width; i++) {
    std::cout << "  " << i << " ";
  }
  std::cout << std::endl;

  std::cout << " |---";
  for (int i = 1; i < m_board_width; i++) {
    std::cout << "|---";
  }
  std::cout << "|" << std::endl;
  for (int i = 0; i < m_board_height; i++) {
    std::cout << i << "| ";
    for (int k = 0; k < m_board_width; k++) {
      if (tiles[i * m_board_width + k].get_value() == 9) {
        std::cout << "M | ";
      } else {
        std::cout << tiles[i * m_board_width + k].get_value() << " | ";
      }
    }
    std::cout << "\n"
              << " |---";
    for (int i = 1; i < m_board_width; i++) {
      std::cout << "|---";
    }
    std::cout << "|" << std::endl;
  }
}

// Revealing the board
void Board::reveal(int row, int col) {
  // checking for coordinate bounds
  if (row < 0 || row >= m_board_height || col < 0 || col >= m_board_width) {
    return;
  }
  // Getting the right index for the tile
  int findindex = row * m_board_width + col;
  Tile &tile = tiles[findindex];

  // if(row, col) revealed
  if (!tile.is_reveal()) {

    // reveal the tile at row,col = set_revealed
    tile.set_revealed(true);

    // if(row, col) is a mine{
    if (tile.get_value() == 9) {
      has_lost = true;
      return;
    }

      revealed_count++;
    // if(row, col) mines have not revealed
    if (revealed_count == ( m_size - minecount) && !has_lost) {
      has_won = true;
      return;
    }
  }
}

  void Board::display_unrevealed(){
    std::cout << " ";
    for (int i = 0; i < m_board_width; i++) {
      std::cout << "  " << i << " ";
    }
    std::cout << std::endl;

    std::cout << " |---";
    for (int i = 1; i < m_board_width; i++) {
      std::cout << "|---";
    }
    std::cout << "|" << std::endl;
    for (int i = 0; i < m_board_height; i++) {
      std::cout << i << "| ";
      for (int k = 0; k < m_board_width; k++) {

        int findindex = i * m_board_width + k;
        if (tiles[findindex].is_reveal()) {
          if (tiles[findindex].get_value() == 9) {
            std::cout << "M | ";
          } else {
            std::cout << tiles[findindex].get_value() << " | ";
          }
        } else {
          std::cout << "# | ";
        }
      }
      std::cout << "\n"
                << " |---";
      for (int i = 1; i < m_board_width; i++) {
        std::cout << "|---";
      }
      std::cout << "|" << std::endl;
    }
  }\[\033[01;34m\]\w\[\033[00m\]$ cat -n Tile.h
// Your Tile.h File
// header file for the Tile class
#ifndef TILE_H
#define TILE_H
#include <iostream>

class Tile {
private:
  int m_value;
  bool m_revealed = false;
```

```
    11
    12  public:
    13  Tile();
    14  void display() const;
    15  void set_revealed(bool reveal);
    16  int get_value() const;
    17  void set_value(int value);
    18  bool is_reveal() const{
    19      return m_revealed;
    20  }
    21  };
    22
    23  #endif\[\033[01;34m\]\w\[\033[00m\]$ cat -n Tile.cpp
     1  // The implementation of the Tile.h file
     2  // Implementation file for the set_revealed and display methods of Tile
     3  #include "Tile.h"
     4  #include <iostream>
     5  Tile::Tile() {}
     6
     7  void Tile::set_value(int value) { m_value = value; }
     8
     9  int Tile::get_value() const { return m_value; }
    10
    11  void Tile::set_revealed(bool reveal) { m_revealed = reveal; }
    12
    13  void Tile::display() const {
    14    if (m_revealed == true) {
    15      if (m_value == 9) {
    16        std::cout << "M";
    17      } else {
    18        std::cout << m_value;
    19      }
    20    } else {
    21      std::cout << "#";
    22    }
    23  }\[\033[01;34m\]\w\[\033[00m\]$ g++ main.cpp Board.cpp Tile.cpp -o minesweeper
\[\033[01;34m\]\w\[\033[00m\]$ ./minesweeper
Enter the difficulty you want to play((width, height, minecount): 3 3 3
   0   1   2
 |---|---|---|
0| # | # | # |
 |---|---|---|
1| # | # | # |
 |---|---|---|
2| # | # | # |
 |---|---|---|
Enter a row and column to dig: 0 0

   0   1   2
 |---|---|---|
0| 0 | # | # |
 |---|---|---|
1| # | # | # |
 |---|---|---|
2| # | # | # |
 |---|---|---|
Enter a row and column to dig: 1 0

   0   1   2
 |---|---|---|
0| 0 | # | # |
 |---|---|---|
1| 0 | # | # |
 |---|---|---|
2| # | # | # |
 |---|---|---|
Enter a row and column to dig: 2 0

   0   1   2
 |---|---|---|
0| 0 | # | # |
 |---|---|---|
1| 0 | # | # |
 |---|---|---|
2| 0 | # | # |
 |---|---|---|
Enter a row and column to dig: 0 1

   0   1   2
 |---|---|---|
0| 0 | 2 | # |
 |---|---|---|
1| 0 | # | # |
 |---|---|---|
2| 0 | # | # |
 |---|---|---|
Enter a row and column to dig: 2 1

   0   1   2
 |---|---|---|
0| 0 | 2 | # |
 |---|---|---|
1| 0 | # | # |
 |---|---|---|
2| 0 | 2 | # |
 |---|---|---|
Enter a row and column to dig: 1 1

YOU've WON!!!
Final Board:
   0   1   2
 |---|---|---|
0| 0 | 2 | # |
 |---|---|---|
1| 0 | 3 | # |
 |---|---|---|
2| 0 | 2 | # |
 |---|---|---|
\[\033[01;34m\]\w\[\033[00m\]$ ./minesweeper
Enter the difficulty you want to play((width, height, minecount): 3 3 3
   0   1   2
 |---|---|---|
0| # | # | # |
 |---|---|---|
1| # | # | # |
 |---|---|---|
2| # | # | # |
 |---|---|---|
Enter a row and column to dig: 0 0

   0   1   2
 |---|---|---|
0| 2 | # | # |
 |---|---|---|
1| # | # | # |
 |---|---|---|
2| # | # | # |
 |---|---|---|
Enter a row and column to dig: 1 0

YOU've LOST!!!!!!!
Final Board:
   0   1   2
 |---|---|---|
0| 2 | M | 1 |
 |---|---|---|
1| M | 3 | 2 |
 |---|---|---|
2| 2 | M | 1 |
 |---|---|---|
\[\033[01;34m\]\w\[\033[00m\]$ exit

Script done on 2023-11-21 22:58:21+00:00 [COMMAND_EXIT_CODE="0"]
```