Minesweeper

The rules of the game are as follows:

There exists a board, which contains a grid of spaces. A space could be a mine, or not. The player clicks on a space, and it gets revealed. The goal of the game is to reveal all the spaces that are not mines, while avoiding the spaces that are.

When a space is revealed:

If it's a mine, the game ends

If it's not a mine, it shows the number of mines adjacent to that space (anywhere from 0 to 8, with 0 just showing as an empty space)

- If a space has no adjacent mines, all non-mine spaces adjacent to it are also revealed The player uses the numbers as clues to figure out where other mines may be located.

When all of the non-mine spaces have been revealed, the player wins!

Flags: Right-clicking a hidden space puts a flag on that space, marking it as a possible mine. Flagged spaces cannot be revealed (with left-clicks or as a result of adjacent spaces being revealed), but another right-click will remove the flag.

Mine Counter: to track the number of mines that are on the board. Every time the player places a flag, the counter goes down by one. Every time they remove a flag, the counter goes up by one. The mine counter CAN go negative!

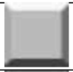
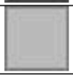
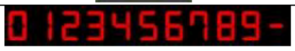
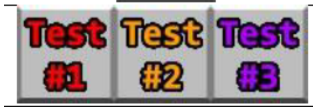
Restart Button: The smiley face centered at the top or bottom of the window lets you restart a new board with everything reset and mines randomized

Non-standard features for your version of this project

Debug Button: Clicking this button will toggle the visibility of the mines on the board. Use this to help you test/debug various features of the game. Having to play the game properly each time you want to test something is very time-consuming. Creating these developer shortcuts helps speed up the development process

Test Buttons #1-3: Another development shortcut, clicking on these loads a file with a specific board layout, details on this later.

Those are the features that your game will need to have. The rules are pretty simple, but even simple games like this can be challenging to implement.

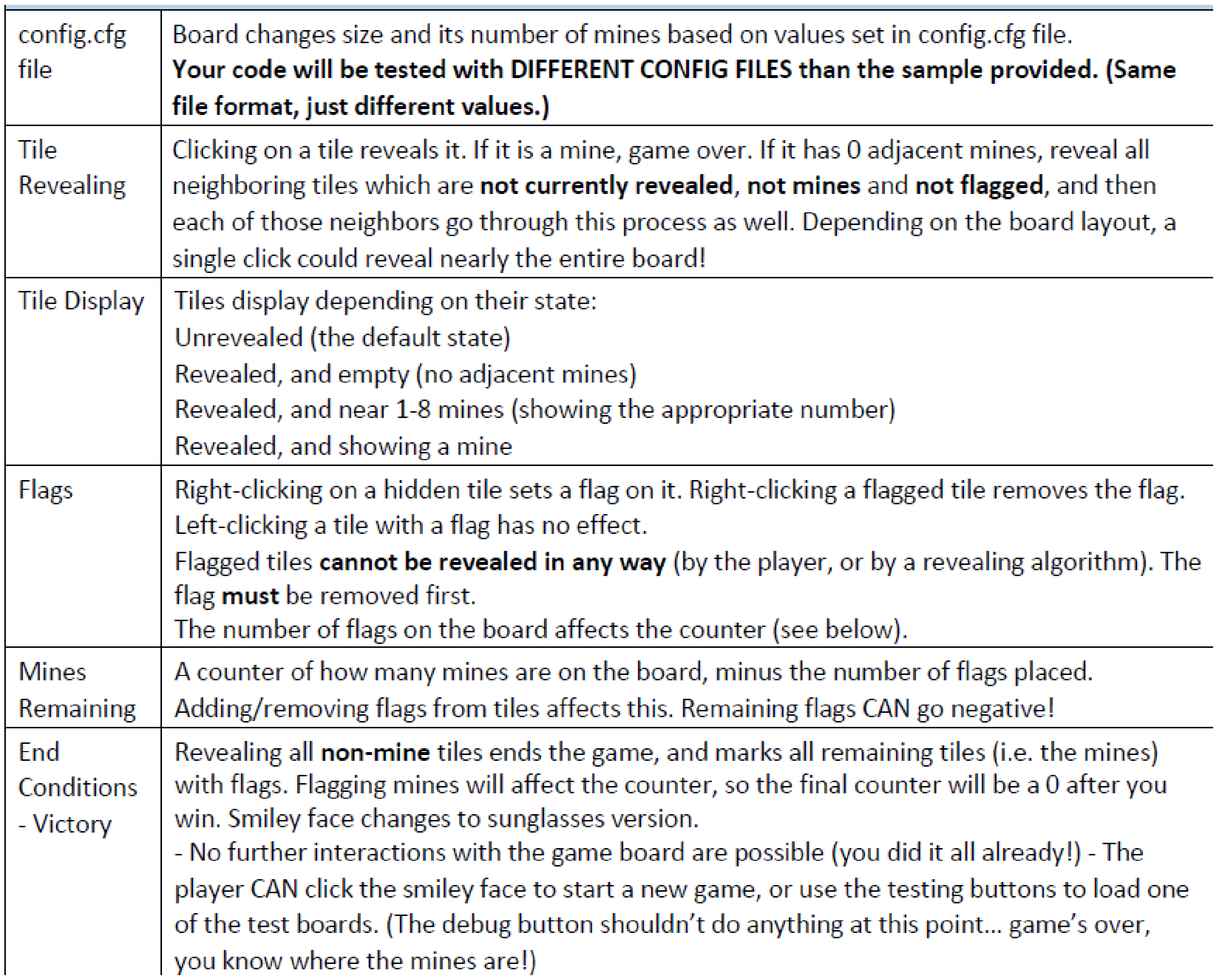
Game Images		
	mine.png	The star of the game (although if you play properly, you'll never see one!)
	tile_hidden.png	What all tiles look like before they are clicked/revealed
	tile_revealed.png	A revealed tile with no adjacent tiles
	number_#.png	Tiles with the numbers 1-8 on them (replace # with the appropriate number. Used for tiles that have 1-8 adjacent mines
	flag.png	Draw this on top of hidden tiles when they are flagged by the player as possible mines.
UI Images		
	face_happy.png	Click this button to reset the map. New mines, everything hidden, it's like you restarted the program.
	face_win.png	Victory!
	face_lose.png	The opposite of victory! (It's cool, no smiley faces were harmed during the creation of this project)
	digits.png	Used for the digits on the "remaining mines" display. You can use this one texture for all the numbers, and change the "texture rect" of a sprite to draw a different portion of the image.
	debug.png	Used to toggle debug mode
	test_1/2/3.png	Used to load test files from which the board will be set

Adjacent Mines and Tiles

In order to calculate the number of nearby mines, as well as when revealing tiles, each tile should store a list of neighboring tiles. A tile could have UP TO 8 neighbors. An easy way to do this is with pointers. Since the number is a variable, a dynamically sized container would be perfect for this. You could also use a fixed-length array, since no tile will ever have more than 8 neighbors.

vector<Tile*> adjacentTiles; // Store each tile near us, the size() of each vector will vary

Tile* neighbors[8]; // Always 8 pointers, some of which might be nullptr



config.cfg file	Board changes size and its number of mines based on values set in config.cfg file. **Your code will be tested with DIFFERENT CONFIG FILES than the sample provided. (Same file format, just different values.)**
Tile Revealing	Clicking on a tile reveals it. If it is a mine, game over. If it has 0 adjacent mines, reveal all neighboring tiles which are **not currently revealed**, **not mines** and **not flagged**, and then each of those neighbors go through this process as well. Depending on the board layout, a single click could reveal nearly the entire board!
Tile Display	Tiles display depending on their state: Unrevealed (the default state) Revealed, and empty (no adjacent mines) Revealed, and near 1-8 mines (showing the appropriate number) Revealed, and showing a mine
Flags	Right-clicking on a hidden tile sets a flag on it. Right-clicking a flagged tile removes the flag. Left-clicking a tile with a flag has no effect. Flagged tiles **cannot be revealed in any way** (by the player, or by a revealing algorithm). The flag **must** be removed first. The number of flags on the board affects the counter (see below).
Mines Remaining	A counter of how many mines are on the board, minus the number of flags placed. Adding/removing flags from tiles affects this. Remaining flags CAN go negative!
End Conditions - Victory	Revealing all **non-mine** tiles ends the game, and marks all remaining tiles (i.e. the mines) with flags. Flagging mines will affect the counter, so the final counter will be a 0 after you win. Smiley face changes to sunglasses version. - No further interactions with the game board are possible (you did it all already!) - The player CAN click the smiley face to start a new game, or use the testing buttons to load one of the test boards. (The debug button shouldn't do anything at this point... game's over, you know where the mines are!)

End Conditions - Defeat	Clicking on a mine ends the game. What should happen: - All tiles with mines are revealed (and display on top of any flags you may have place) - The smiley face changes to the dead face (he's just acting, don't worry!) - No further interactions with the game board are possible. - The player CAN click the dead smiley face to start a new game, or use any of the testing buttons. (The debug button shouldn't do anything at this point... game's over, you know where the mines are!)
Random Mine Placement	When the game **starts** and when the board is **reset** (by clicking the smiley face button), a number of mines equal to the number specified in the .cfg file are randomly placed on the map--no more, no less.

Test Buttons (1, 2 and 3)	3 test buttons to load testBoard1.brd, testBoard2.brd, testBoard3.brd. Each button updates all board spaces according to the data in the file, resets all flags and the counter – it's like you just started the program, with a specific set of data instead of random tiles. **Your code will be tested with DIFFERENT BOARD FILES than the samples provided. (Same file format, just different 1s and 0s—don't hard-code results!)**
Debug Button	Clicking the debug button toggles whether or not to show the mines on the map. Since the intent of this feature is to help YOU, the programmer, see the mines, draw the mines OVER anything else. Don't stop drawing anything else. These are in addition to whatever is normally being displayed.