\*\* REMINDER: Your programs MUST valgrind cleanly for full credit! \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

For this problem, you will be completing a partially written

minesweeper game. This game will be played on a text interface (not a

GUI--we haven't learned anything about those).

I have provided code for an almost working minesweeper in

minesweeper.c, except that we have deleted the code for 3 functions:

board\_t \* makeBoard(int w, int h, int numMines)

int countMines(board\_t \* b, int x, int y)

void freeBoard(board\_t \* b)

Your job is to implement each of these functions (which all have a

//WRITE ME comment to make them easy to find). A brief description of

each follows:

- makeBoard: this function should malloc and initialize a board\_t

representing the board. The parameters specify the width and

height of the board, as well as the number of mines. You will also

need to call malloc (multiple times) to allocate space for the 2D

array "board". This function should fill in all squares on the

board with UNKNOWN, then call addRandomMine an appropriate number

of times (i.e., numMines) to "randomly" place mines on the

board. Note that the fields of your board\_t must be initialzed

before you call addRandomMine. Also note that the mine generation

is pseudorandom and will not change if you re-run the program

multiple times with the same parameters.

Note that the layout of b->board should be such that it is indexed

b->board[y][x]

where y is between 0 and the height and x is between 0 and the

width.

- countMines: this function takes a board\_t, and an (x,y)

coordinate. It should count the mines in the 8 squares around that

(x,y) coordinate and return that count. Note that a mine may be

indicated by a square on the board either being HAS\_MINE or

KNOWN\_MINE. You can use the IS\_MINE macro to test both cases:

IS\_MINE(b->board[ny][nx])

(where b is the board\_t, and (nx,ny) are the coordinates you want

to check). Be careful not to go out of bounds of the array.

- freeBoard: This function takes a board\_t and frees all memory

associated with it (including the array inside of it). That is,

freeBoard should undo all the allocations made by a call to

makeBoard.

Note: You should NOT change any of the other provided functions!

Once you have these all working, you should have a playable game of

minesweeper. Note that there are a few differences in game play from

the "standard" game:

- You select a square by entering its x (column) and y (row)

coordinate. The x coordinates are listed across the top and the y

are listed down the left side to reduce counting.

- The game will automatically figure out the "obvious" squares: both

mines and non-mined spaces. It will reveal these to you as soon as

it considers them trivial to figure out.

- You cannot manually mark a square that you suspect has a mine.

Once your code is complete, submit minesweeper.c.