CSE1PES: PROGRAMMING FOR ENGINEERS AND SCIENTISTS

ASSIGNMENT 3

TOOLS

- Unix server through Putty — for compiling and testing
- Notepad ++ — for writing solutions
- LMS — for submission
- Microsoft word (or similar) — for writing written answers

SUBMISSION

The task is split into two submissions:

- **The code**
  - **Submitted as a .c file in the normal assignment submission portal. (Do not submit the executable.)**
- **The written portion**
  - **Submitted as a PDF to the Turnitin assignment portal.**

Each has a separate submission portal, ensure to submit both.

There are NO late submissions for this assignment. No assignment will be accepted after the due date, without a legitimate extension granted by the lecturer before the due date.

ACADEMIC INTEGRITY

Plagiarism is the submission of somebody else's work in a manner that gives the impression that the work is your own. For individual assignments, plagiarism includes the case where two or more students work collaboratively on the assignment. The School of Engineering and Mathematics treats plagiarism very seriously. When it is detected, penalties are strictly imposed.
http://www.latrobe.edu.au/students/academic-integrity

MINESWEEPER

BACKGROUND

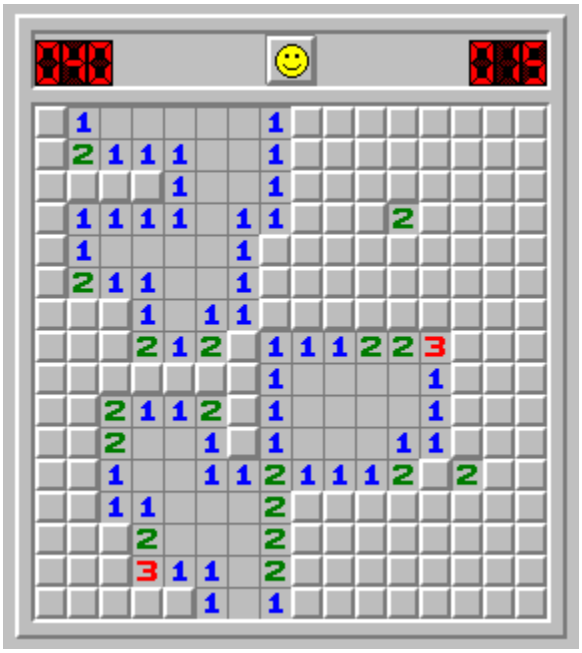


Image source: www.freeminesweeper.org

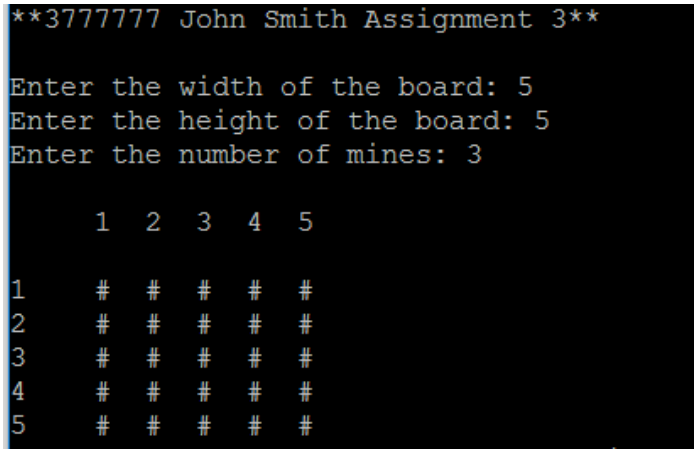
Please watch this video for the rules of minesweeper:
https://www.youtube.com/watch?v=93oSIfWN0HU

To try a game:
http://minesweeperonline.com/

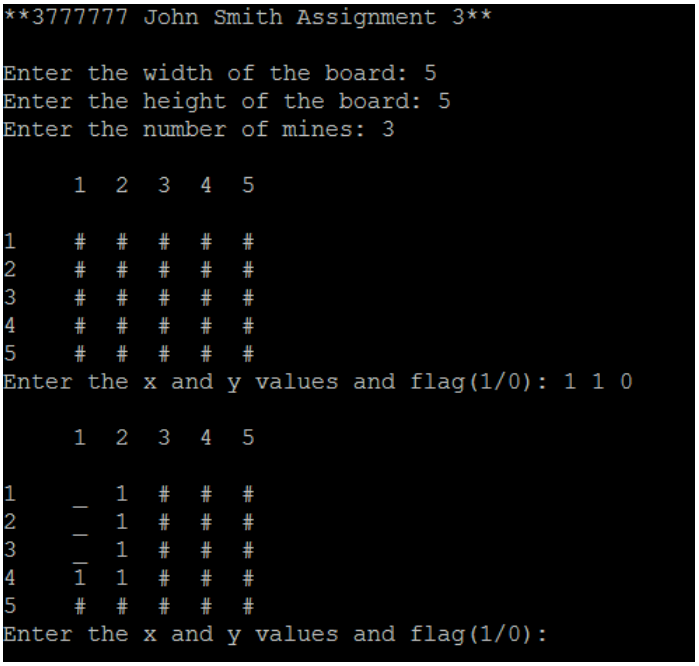
PROBLEM

In this problem we want to create a playable game of minesweeper.

This game will run by allowing a user to choose the size of the board, the number of mines, coordinate value to select and select if they wish to flag a coordinate. The game starts as an unrevealed matrix as shown below:



If a coordinate with no mines surrounding it is pressed, then all other coordinates that surround that have no mines (and so on) will be revealed as shown below:



Please note this is not the ideal solution, but one designed to test your understanding of various topics.

STAGES

To ensure that you can complete as much of this assignment to the best of your ability, the assignment is set up in three stages.

Stage 1: Creating both the initialization functions, the printing functions, structs, macros, global variables and first **nine** steps of main.

Stage 2: Creating the make_move and get_move functions, and all of main

Stage 3: Create the uncover_zeros function

FUNCTION PROTOTYPES, GLOBAL VARIABLES AND INCLUDES

1. You must include the stdio.h and stdlib.h libraries
2. The following function prototypes are to be used:

```
void initialize_mine_board(struct Game game, struct Location** mine_board);
void initialize_mine_values(struct Game game, struct Location** mine_board);
void make_move(struct Game game, struct Location** mine_board, int x, int y);
void get_move(int* x_ptr, int* y_ptr, int* flag_ptr, int max_x, int max_y);
void print_mine_board(struct Game game, struct Location** mine_board);
void uncover_zeros(struct Location** mine_board, int x, int y);
```

   a. All are described below
   b. STAGE 1: Implement the first three functions from above and complete the below steps, then test up to step 9 in main.
   c. You **can** change the naming
   d. You **cannot** change the types
3. The following structs are to be used

```
struct Location
{
    int value;
    char status_byte;
};

struct Game
{
    int width;
    int height;
    int mines;
    int mines_flagged;
    int flags;
};
```

4. Create a **global** variable which is of type int, and initialize to 1. This will track if the player has still not clicked on a mine. When selecting a mine, this will change to value 0 to indicate the game is over.
5. Define a macro named MINE as value 0, FLAG as value 1, and REVEALED as value 2. These are to be used as bit flags for the status_byte variable in struct Location.

    The macros values represent the bit position in the status_byte, as follows:
    MINE: If 1, indicates that the particular location contains a mine
    FLAG: If 1, indicates that the player has flagged this particular location
    REVEALED: If 1, indicates that the player has selected this position and the contents (value, mine or blank) of the location are visible/revealed.
    (Note: For REVEALED and FLAG, they both cannot be 1 for a particular location)

FUNCTION - MAIN

1. The program prints to the screen the student number, student name and the assignment number. This is enclosed by double asterisks. Ensure you follow the below format.

```
**3777777 John Smith Assignment 3**
```

You probably want this to say Assignment 3

2. Create the following variable as shown below:

```
struct Game game;
```

3. Set both the mines_flagged and flags variable in game (struct) to 0
4. Ask the user for the width and height of the board, and store in the game variable appropriately.

```
Enter the width of the board: 5
Enter the height of the board: 5
```

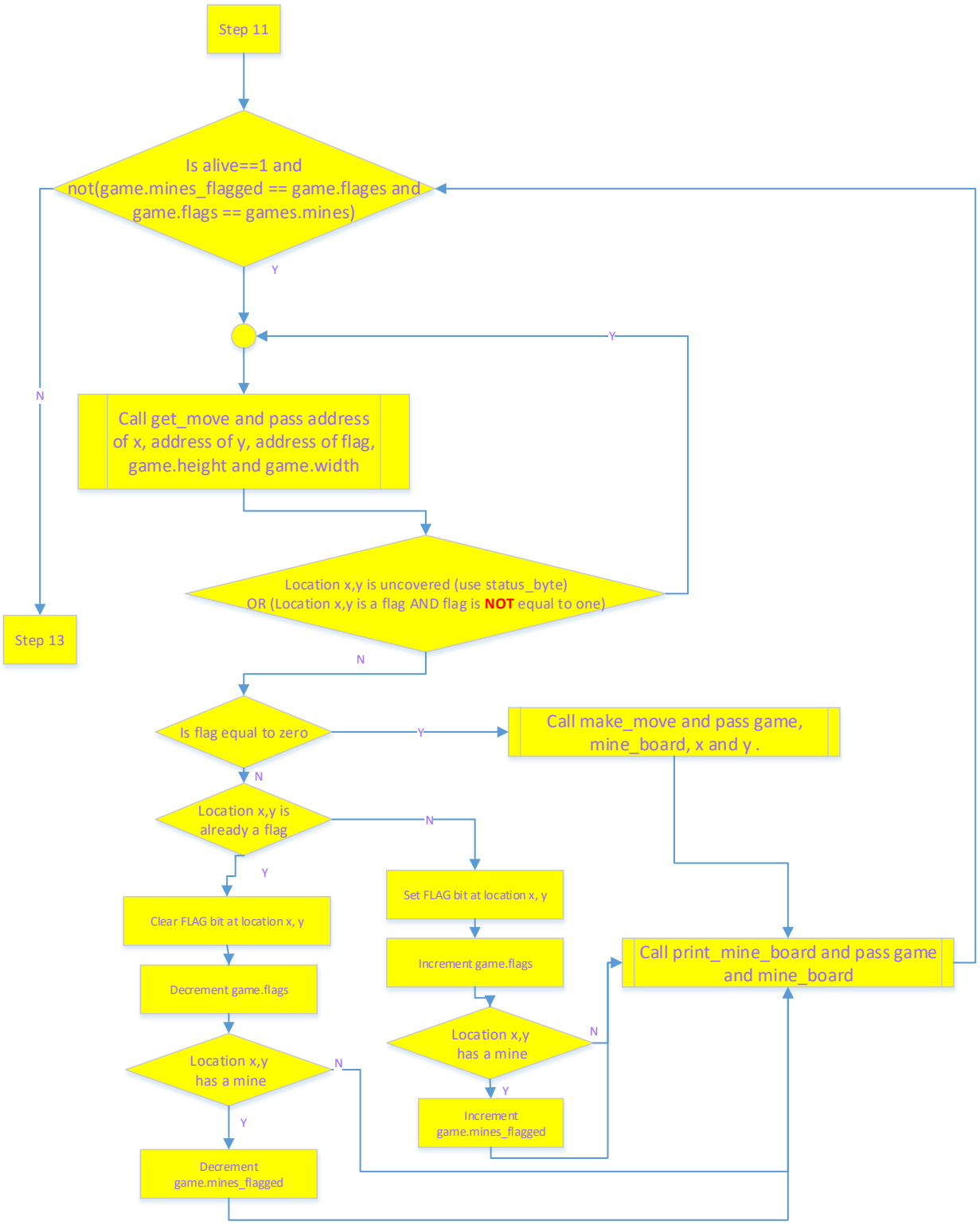
5. Ask the user for the number of mines, and store in the game variable appropriately. You should also ensure that the number of mines is less than the product of the width and length. If it is not, then keep asking the user to enter until it is. (One line invalid message ending in newline, you can choose an appropriate message)
6. Create a pointer to a pointer to a struct Location named 'mine_board' and dynamically allocate memory of size game.height+2 multiplied with the size of struct Location* . *(This is the rows of your 2D array) (Explanation of this will be provided on LMS)*
7. Iterate from 0 to game.height+2 :
   And allocate pointers to struct Location s as shown: (iteration not shown)

```
mine_board[i] = (struct Location*)malloc(((game.width)+2)*sizeof(struct Location));
```

(This is the columns of your 2D array)

8. The program then initializes the board using the appropriate function
9. The program then prints the board to the screen using the appropriate function
10. STAGE 1: Stop here and check your program works (AND SAVE A COPY). Make sure you have implemented the first 3 functions listed above.

11. Declare variables for the flag, x location and y location as appropriate types.
12. Follow the flow diagram to write the next segment of code



13. Check the alive variable to test if the user won or lost. Print an appropriate message to the user. The message is up to you (Refer to bonus).
14. End the function appropriately *(Hint. Free dynamically allocated memory)*

FUNCTION – INITIALIZE_MINE_BOARD

1. Create iteration variables i and j
2. Iterate from 0 to game.height+1 (inclusive) using i :
    a. Iterate from 0 to game.width+1 (inclusive) using j :
        i. Set status_byte to 0 for each location on mine_board
3. Create a variable temp_mines and initialize to game.mines
4. Create variables x and y
5. Create a loop that continues while temp_mines is greater than 0
    a. Set the x and y values to random locations as follows:

```
y=rand()%game.height+1;
x=rand()%game.width+1;
```

    b. Test the MINE bit in the status_byte given the x and y coordinates is 0

```
if(!(mine_board[y][x].status_byte&(1<<MINE)))
```

    c. If so, set the MINE bit in the status_byte given the x and y coordinates and decrement temp_mines by one.
6. Call the initialize_mine_values function

FUNCTION –INITIALIZE_MINE_VALUES

1. Create iteration variables i and j
2. Iterate from 0 to game.height+1 (inclusive) using i :
    a. Iterate from 0 to game.width+1 (inclusive) using j :
        i. **If** i is equal to 0 or i is equal to game.height+1 or j is equal to 0 or j is equal to game.width+1 then set the value of the i,j coordinate in mine_board to 0 as shown

```
mine_board[i][j].value=0;
```

        and the REVEALED bit in the status_byte to 1 (Using bit manipulation)
        ii. **Else** create a variable named count
            1. Count the number of mines by checking the MINE bit of all the locations that surround the location i and j (Hint: there are 8 checks that need to be done)
            2. Set the value in value for the location to count

```
mine_board[i][j].value=count;
```

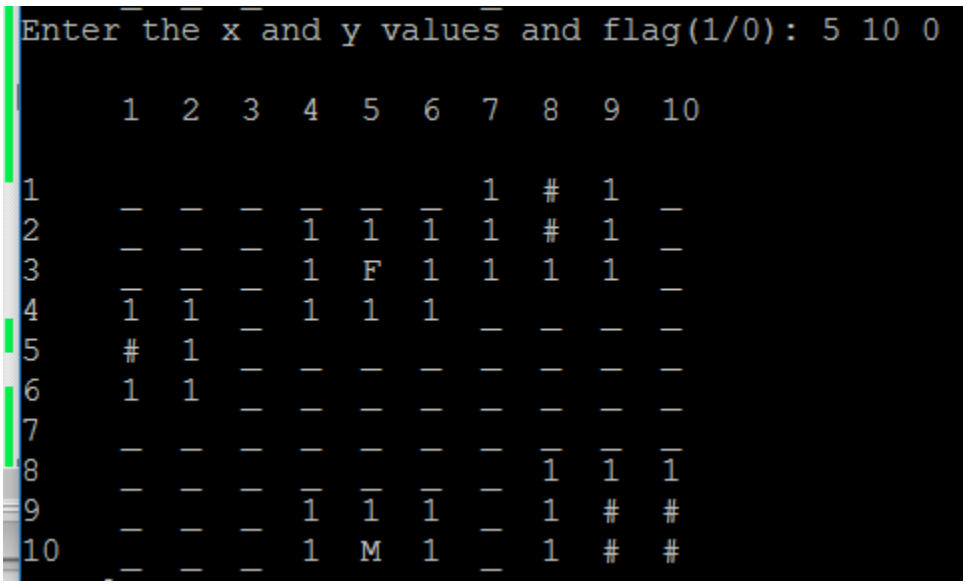
FUNCTION – PRINT_MINE_BOARD

Print the board such that it prints the x and y numbers around the board.

Iterate through each location (1 to game.height, 1 to game.width) (Ignore locations 0 and game.height+1, game.width+1) :

If the FLAG bit is set then print a 'F'
else if the REVEALED bit is set then
if the MINE bit is set print a 'M'
else if the value is 0 print a '_'
else print the value
else print a '#' to indicate the location is covered

(Note: There is a space before and after each location.)



FUNCTION – MAKE_MOVE

This function takes the game information, the board information and the location of the coordinate to be revealed. It tests if the coordinate is a mine or not, and if it is then set the appropriate variable to end the game.

Steps:

1. If the coordinate had a mine, set the appropriate bit to revealed for the given coordinate and set the variable that tracks if the player has lost to an appropriate value
2. Else set the appropriate bit to revealed for the given coordinate and
    a. If a flag, unset the flag bit and handle the game variables appropriately (hint. you have done this once already in main)
    b. If the value of the coordinate is 0 then
    STAGE 2: Leave blank, come back and add the uncover_zeros call here when you get to STAGE 3 (SAVE A COPY)

FUNCTION – UNCOVER_ZEROS

Create the function that:

1. Uncovers all surrounding coordinates (there are 8)
2. If any of the surrounding coordinates are of value 0, that are not revealed and that are not flags, then repeat either recursively or iteratively.

This function should uncover groups of 0 values, refer to instruction video.

(Note: You do not need the size of the board to do this function)
(Note: This function is not very long 10 – 30 lines if done correctly and efficiently)

FUNCTION – GET_MOVE

Create the function that gets a move from a player, that takes a pointer to the x coordinate, y coordinate and flag; and the max_x coordinate and max_y coordinate.

Ask the user for the x, y and flag (yes to flag is 1 and no to flag is 0)
Store in the location pointed to by the pointers appropriately.
Continue asking the user if locations are invalid (x or y is less than 1, x or y is greater than respective max, flag is not 0 or 1)

CONSTRAINTS

- Text to user needs to be easily understandable. You can change the text but the same inputs must be used.
- The program must print your student number, name and the assignment number as specified.
- Types should be used appropriately.
- You must use comments to explain significant lines of code. Function descriptions must be included. Program description must be included

HINTS

- The code should be around 250 – 350 lines of code without the bonus (without comments). If your code is **significantly** larger you may want to reconsider your approach.

SUBMISSION

Part 1. Solve the problem by implementing a program using C code.

Part 2. Produce a report of 500 +/- 250 words:

- 1. Explain how you implemented "FUNCTION – GET_MOVE", why you implemented in this way.
- 2. Explain how step 6 and 7 in "FUNCTION – MAIN" is implemented, why it is implemented in this way and how memory is allocated in this step.
- 3. Explain what happens when the 'make move' function is called, how the variables are passed and how they are returned. Use a diagram (box and pointer or similar) to help explain.

EXAMPLES

Given on LMS, assignment tab

BONUS

Given on LMS, assignment tab

RUBRIC

Part 1 – 86 points				
Is the student's number, student name and assignment number printed?	ACADEMIC INTEGRITY If the names are different	0 Incorrectly formatted	-	1 Correctly formatted
Does the program compile?	0 Does not compile.	1 Compiles with minor errors or many warnings.	2 Compiles with a warning.	4 Compiles appropriately.
Is the code indented correctly?	0 Not indented or poorly indented	1 Any mistake in indenting		2 Indented correctly
Are the inputs and outputs understandable?	0 Does not accept inputs correctly or does not output data appropriately.	1 Some errors in outputs or inputs.	2 Minor errors in outputs or inputs.	3 Inputs and outputs are handled appropriately.
Suitable comments are used to explain particular blocks or lines of code.	0 No comments	4 Some comments used throughout.	6 Reasonable comments used throughout.	8 Excellent descriptions and commenting.
Is the printing of the board correct?	0 The tables do not use any formatting.	2 The board is formatted but incorrectly.	4 Minor errors in formatting	6 Correct format specifiers, numbering, spacing and new lines are used
STAGE 1: Variables created and initialized correctly	0 Was not implemented	2 Was implemented with major errors	4 Was implemented with minor errors	6 Was implemented
STAGE 1: Initializes the mines and values correctly	0 Was not implemented	2 Was implemented with major errors	4 Was implemented with minor errors	6 Was implemented
STAGE 1: Initializes the values correctly	0 Was not implemented	2 Was implemented with major errors	4 Was implemented with minor errors	6 Was implemented
STAGE 1: Initializes the board correctly.	0 Was not implemented	2 Was implemented with major errors	4 Was implemented with minor errors	6 Was implemented
STAGE 2: Implementation of the flow diagram and game ending.	0 Was not implemented	2 Was implemented with major errors	4 Was implemented with minor errors	6 Was implemented

STAGE 2: Implementation of make_move function.	0 Was not implemented	2 Was implemented with major errors	4 Was implemented with minor errors	6 Was implemented
STAGE 2: Implementation of bitwise operations	0 Was not implemented	2 Was implemented with major errors	4 Was implemented with minor errors	6 Was implemented
STAGE 2: Implementation the get_move function	0 Was not implemented	2 Was implemented with major errors	4 Was implemented with minor errors	6 Was implemented
STAGE 3: Implementation of uncover_zeros function	0 Was not implemented	6 Was implemented with major errors	8 Was implemented with minor errors	10 Was implemented
Are appropriate variable types and names used where appropriate? And concepts taught used appropriately?	0 All variables are declared as the largest types. Concepts taught are used inappropriately	1 Most types or names are not suitable. Or concepts taught are used but are not suitable.	2 Some types or names are not suitable. Concepts taught are used incorrectly.	4 All variable types and names are suitable. Concepts taught are used suitably and appropriately.
Part 2 – 14 points				
Is the student's number and student name on the report?	ACADEMIC INTEGRITY If the names are different	-	-	0 Included
Question 1	0 Poor explanation.	2 Reasonable explanation.	3 Good explanation.	4 Excellent explanation.
Question 2	0 Poor explanation.	2 Reasonable explanation.	3 Good explanation.	4 Excellent explanation.
Question 3	0 Poor explanation.	2 Reasonable explanation.	3 Good explanation.	4 Excellent explanation.
Does the explanation use appropriate grammar and spelling?	0 Poor grammar or many spelling mistakes	1 Some errors in grammar or spelling.	-	2 Reasonable grammar and spelling.