CAB403 Assignment Marking Criteria

Student Name(s): Koh Kim Hai & Jeff Lo

Student Number(s):N9329013 & N9620699

Total Marks: _____ /**100**

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Statement of Completeness

Fully Implemented
Partial Implementation
Non-Implementation

All tasks were done equally together.

Task 1: (____/50)

Criteria	Marks
Network byte order is used for transmitting multi-byte data types (server and client)	/2
Server command line parameter – configurable port & default port	/2
Server authenticates client using data in *"Authentication.txt"* file	/5
Server exits gracefully upon receiving SIGNAL (ctrl + c)	/3
Client command line parameters	/2
Client menu implementation	/3
Client exits gracefully when user selects "Quit" option from menu	/2
Quitting part-way through a game resets the playfield	/2
Random number generator is seeded as per assignment specification	/1
Mines are placed on playfield using algorithm from assignment specification	/1
Revealing a tile with adjacent mines shows the number of adjacent mines	/2
Revealing a tile with zero adjacent mines reveals neighbouring zero mines	/4
Revealing a tile with a mine ends the game, and displays the full playfield	/2
Placing a flag on a tile with a mine decrements the remaining mine count	/2
Placing a flag on a tile without any mine has no effect on remaining count	/2
Placing flags on all mines results in a game win, and the full playfield displayed	/4
Time taken to complete the game is measured and displayed correctly	/2
Leader board is updated after winning a game	/3
Leader board is displayed as per assignment specifications	/3
Description of the data structure that is used for the Leader Board in your report	/3

Task 2: (____/20)

Criteria	Marks
Multithreaded implementation	/10
Process synchronization	/6
Description of how the critical-section problem is handled in your report	/4

Task 3: (____/16)

Criteria	Marks
Thread pool creation	/7
Thread pool use	/3
Thread pool cleaning	/3
Description of how the thread pool is created and managed in your report	/3

~~PROGRAM QUALITY~~

Marks: (____/10)

Criteria	Marks
Program structure & readability	/1
Program performance	/2
Resource management	/3
Program reliability (e.g. run time errors, deadlocks, file I/O)	/3

REPORT QUALITY

Marks: (____/4)

Criteria	Marks
Statement of completeness	/2
Instructions on how to compile and run your program	/2

Comments:

NOTE: Allocation of marks depends on which tasks are attempted.

Task	Functional and Non-functional Requirements	Code Quality	Report	Maximum Marks
Task 1 only	50	10	4	**64**
Task 1 and Task 2	70	10	4	**84**
Task 1, Task2 and Task 3	86	10	4	**100**

Known Bugs and Issues:

1. Choosing to reveal (7,8) instead reveals (7,0)
2. Choosing to reveal (1,8) wrongly recursively reveals its neighbours
3. Stepping on mines do not correctly display the mine locations as in figure 10 of the assignment sheet.
4. Placing a flag at (7,1) does not defuse the mine that is there
5. Entering a non-exact input as specified will send the program into an infinite loop.

Description of Data Structures:

Playfield

GameState

The GameState struct contains integers that act as Booleans in C because C does not have Booleans
These fields are: game_quit, game_won, game_over;

There is a field of type Double to store the time taken to complete the game
The field is: time_taken;

There is a field of type Int to store the number of available mines still in the game.
The field is: current_mines_active;
This number initialises at NUM_MINES

There is a field of custom type Tile to store the 2D array of tiles to contain the tile information.
The field is: tiles[NUM_TILES_Y][NUM_TILES_X];

Tile

The custom type Tile is a struct that contains integers that act as Booleans.
These fields are: is_revealed, is_mine;

There is a field of type Int to store the number of mines that surround this tile.
This field is : adjacent_mines;

There is a field of type char to store a symbolic representation of the tile.
The field is: display_symbol;
If the tile is a mine, it will display a '*'
If the tile is not a mine, it will display the number of surrounding mines (0-8);

Leaderboard

this section has not yet been implemented

Critical-Section

this section has not yet been implemented

Threadpool

this section has not yet been implemented

Instructions on how to compile and run the program.

Code Compilation:

1. Unzip the folder "n9620699_n9329013"
2. Open up a terminal console and navigate to the unzipped folder using the command:" cd Downloads/n9620699_n9329013"

To Compile the Server:

3. Enter the command:" gcc -o server server.c -lpthread" into the terminal console

To Compile the Client:

4. Enter the command:" gcc -o client client.c" into the terminal console

To Run the Server Code

5.1.1. Run the server code with the command:". /server {Unique PORT_ID}"

5.1.2. To close the server, use the SIGINT command with the keyboard inputs "ctrl+c"

No further inputs to the Server Code is necessary.

To Run the Client Code

5.2.1. Run the client code with the command:" ./client {IP_ADDR} {Unique PORT_ID}"

5.2.2. You will be prompted for a ***username***. Enter the username as required. IE. {Maolin} can be entered. Only ONE input is required and will be accepted. More than ONE input will cause the connection to the server to be terminated.

5.2.3. You will then be prompted for a ***password.*** Enter the password as required. IE. {111111} can be entered. Only ONE input is required and will be accepted. More than ONE input will cause the connection to the server to be terminated.

5.2.3.1. Upon unsuccessful login to the server (incorrect username AND/OR password) will result in the connection to the server being terminated.

5.2.3.2. Upon successful login to the server, a welcome message will be displayed. The user can then choose from THREE options: [1. Play Minesweeper, 2. View Leaderboard, 3. Quit.]. The user can input ONE choice into the console with the command:" {OPTION_NUMBER}".

5.2.3.2.1. Entering {3} into the terminal console will cause the connection to the server from the client to be ***CLOSED***. The ***PORT_ID*** will now be available for another potential connection.

5.2.3.2.2. Entering {2} into the terminal console will display the current available leader board.

5.2.3.2.2.1. If the leader board is empty, a text will notify the user. The user can then choose from the same THREE options in 2.2.3.2.

5.2.3.2.2.2. If the leader board is **not** empty, the user will be able to see the standings. The user can then choose from the same THREE options in 2.2.3.2

5.2.3.2.3. Entering {1} into the terminal console will start a game of Minesweeper™. The game board will then be displayed. This game board should be empty. The user can then choose form THREE NEW options: [R. Reveal Tile, P. Place Flag, Q. Quit]. The user can input ONE choice into the console with the command:"{OPTION_LETTER}".

5.2.3.2.3.1. Entering {Q} into the terminal console will display a prompt to ask the user if the user really wishes to quit the game. The user can then

choose from TWO options [Y/n]. The user can then enter ONE choice into the console with the command: "{OPTION_LETTER}".

5.2.3.2.3.1.1. Entering {N} into the terminal console will display the current map and the same THREE options in 2.2.3.2.3.

5.2.3.2.3.1.2. Entering {Y} into the terminal console will cause the game over condition to be met and this will be updated in future leader boards entries.

5.2.3.2.3.2. Entering {P} into the terminal console will display a prompt to ask the user for the co-ordinates to place the flag. The user is then expected to enter TWO values (x→[column index], y→[row index]).

5.2.3.2.3.2.1. The client will then send these arguments to the server to be processed and an updated map will be displayed. The user will then be returned to an input state like 2.2.3.2.3 with the updated map.

5.2.3.2.3.2.1.1. If the tile is not a mine, a text will be displayed to inform the player to choose another tile to place a flag on.

5.2.3.2.3.2.1.2. Else if the tile is a mine, the mine will be diffused, and the total available mines will decrement by 1.

5.2.3.2.3.3. Entering {R} into the terminal console will display a prompt to ask the user for the co-ordinates to reveal the tile. The user is then expected to enter TWO values (x→[column index], y→[row index]).

5.2.3.2.3.3.1. The client will then send these arguments to the server to be processed and an updated map will be displayed. The user will then be returned to an input state like 2.2.3.2.3 with the updated map.

5.2.3.2.3.3.1.1. If the tile is a mine, the game over condition will be met and the game will be updated like 2.2.3.2.3.1.2

5.2.3.2.3.3.1.2. Else if the tile is not a mine but is surrounded by a non-zero number of neighbouring mines, it will display the number of its adjacent mines.

5.2.3.2.3.3.1.3. Else if the tile is not a mine but is surrounded by exactly ZERO neighbouring tiles. It will recursively reveal its neighbouring tiles according to the rules set out in the above subpoints. The recursive function is expected to reveal neighbouring tiles until it runs out of tiles not surrounded by mines.

5.2.3.2.4. After the various inputs, the game is expected to trigger only ONE of the game over, quit game or game won conditions. These will break the break out of its while loop to implement the cleanup() function and process the user statistics for the leader board.

5.2.3.2.4.1. In the cleanup() function, memory allocated to the game and the thread is freed up for other use.

5.2.3.2.5. From here, the user will be returned to the welcome menu like in 2.2.3.2, where the user can choose from the same options and start all over again.