Project 3 - Minesweeper Game

CS 0447 — Computer Organization & Assembly Language

Check the Due Date on the CourseWeb

The purpose of this project is for you to practice writing MIPS assembly program with functions. Your goal is to initialize a Minesweeper game and read mouse input.

Mouse Behavior

During a game play, a cell can be in one of these four states:

- closed without a flag on top of it,
- closed with a flag on top of it,
- open with a number in it, or
- open with nothing in it.

If a cell display something else (e.g., a mine), it means the game is just over. Note that the behavior of a mouse click on a cell will be based on what is currently display on the cell.

Left-Click on a Cell

The following are what the game should do when a user perform a **Left-Click** on a cell:

- **Closed without a flag on top of it**: Recursively open it. To understand the recursively open procedure, assume that you have a two dimensional variable named `board` and the array at 0xffff8000 is the two dimensional array named `mine`. Assume that you are already initialized the array `board` with mines and numbers (previous project). The idea is that if the closed cell contains a mine, that mine explodes and the game is over. If the closed cell contains a number, simply display that number. But of the closed cell contains nothing, display nothing at that cell and open all closed cells around it. Here is a pseudo code for this procedure:

```
int recOpen(int row, int column)
{
    if board[row][column] contains a mine
        display all mines on the Minesweeper board
        set mine[row][column] to 12 (exploded mine)
        return 1
    else if board[row][column] contains a number
        display that number at mine[row][column]
        return 0
```

```
    else if board[row][column] contains nothing
    {
        display nothing (9) at mine[row][column]

        int returnValue;

        if row - 1 and column - 1 are inbound
            returnValue = recOpen(row - 1, column - 1);
            if returnValue == 1, return 1;
        if row - 1 and column are inbound
            returnValue = recOpen(row - 1, column);
            if returnValue == 1, return 1;
        if row - 1 and column + 1 are inbound
            returnValue = recOpen(row - 1, column + 1);
            if returnValue == 1, return 1;
        if row and column - 1 are inbound
            returnValue = recOpen(row, column - 1);
            if returnValue == 1, return 1;
        if row and column + 1 are inbound
            returnValue = recOpen(row, column + 1);
            if returnValue == 1, return 1;
        if row + 1 and column - 1 are inbound
            returnValue = recOpen(row + 1, column - 1);
            if returnValue == 1, return 1;
        if row + 1 and column are inbound
            returnValue = recOpen(row + 1, column);
            if returnValue == 1, return 1;
        if row + 1 and column - 1 are inbound
            returnValue = recOpen(row + 1, column + 1);
            if returnValue == 1, return 1;

        return 0;
    }
}
```

- **Closed with a flag on top of it**: Do nothing
- **Open with nothing in it**: Do nothing
- **Open with a number it it**: If the number of flags around the clicked cell is **not** the same as the number in the cell, do nothing. Otherwise, open around the clicked cell. If a cell is currently close, use the `recOpen` above. If the cell is open, move on to the next cell. The pseudo code is shown below:

```
int openAround(int row, int column)
{
    int returnValue;

    if row - 1 and column - 1 are inbound
```

```
        if mine[row - 1][column - 1] is closed
            returnValue = recOpen(row - 1, column - 1);
            if returnValue == 1, return 1;
    if row - 1 and column are inbound
        if mine[row - 1][column - 1] is closed
            returnValue = recOpen(row - 1, column);
            if returnValue == 1, return 1;
    if row - 1 and column + 1 are inbound
        if mine[row - 1][column - 1] is closed
            returnValue = recOpen(row - 1, column + 1);
            if returnValue == 1, return 1;
    if row and column - 1 are inbound
        if mine[row - 1][column - 1] is closed
            returnValue = recOpen(row, column - 1);
            if returnValue == 1, return 1;
    if row and column + 1 are inbound
        if mine[row - 1][column - 1] is closed
            returnValue = recOpen(row, column + 1);
            if returnValue == 1, return 1;
    if row + 1 and column - 1 are inbound
        if mine[row - 1][column - 1] is closed
            returnValue = recOpen(row + 1, column - 1);
            if returnValue == 1, return 1;
    if row + 1 and column are inbound
        if mine[row - 1][column - 1] is closed
            returnValue = recOpen(row + 1, column);
            if returnValue == 1, return 1;
    if row + 1 and column - 1 are inbound
        if mine[row - 1][column - 1] is closed
            returnValue = recOpen(row + 1, column + 1);
            if returnValue == 1, return 1;

    return 0;
}
```

Right-Click on a Cell

The following are what the game should do when a user perform a **Right-Click** on a cell:

- **Closed without a flag on top of it**: Put a flag on top of it
- **Closed with a flag on top of it**: Remove a flag from it
- **Open with nothing or a number in it**: Do nothing

What To Do?

The main goal is to write the MIPS assembly program to drive this Minesweeper game so that it behaves like an actual Minesweeper game.

Part I: Open All Cells and Close them (10 Points)

When the program starts, reset, or the size is changed, your program should open all cells. Then display the message "`Press Enter key to continue...`" on the console screen and wait for a keyboard input using read string system call (`syscall 8`). Once user press the Enter key, close all cells and wait for mouse command.

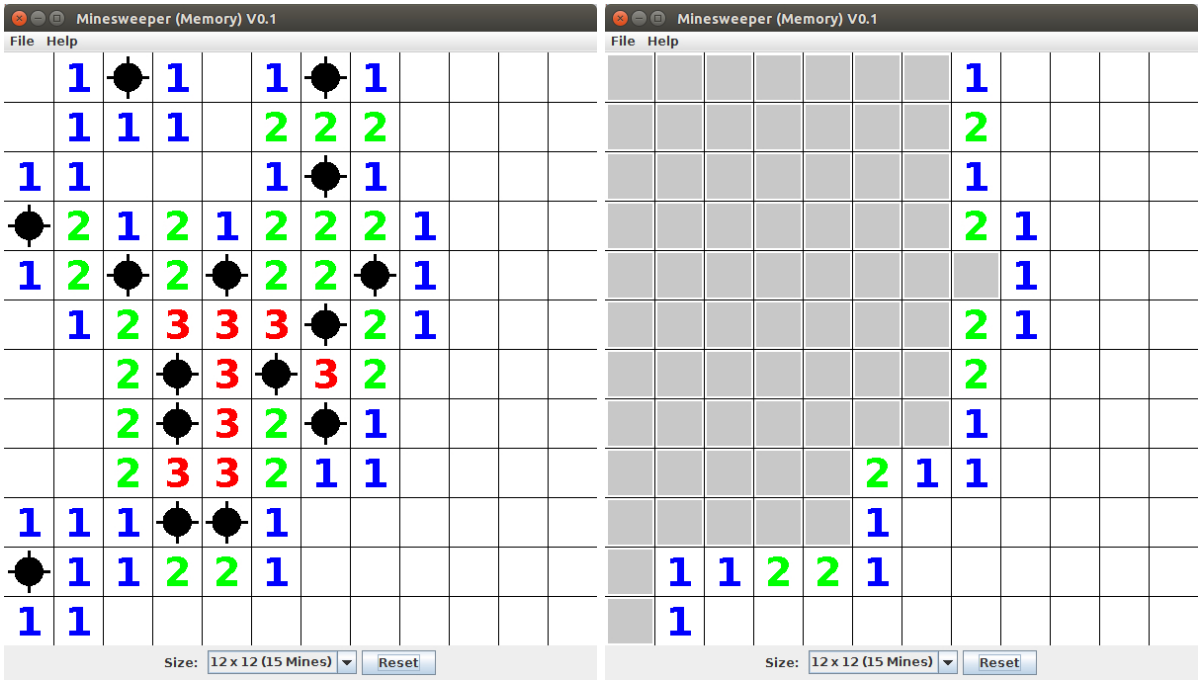
Part II: Simple Mouse Click (30 Points)

For this part, we are going to check your mouse click ability as follows:

- (5 points) Left click on a close cell that contains a number: Open that cell with the number
- (5 points) Left click on a close cell that contains a mine: Display all mines on the Minesweeper hardware and display the exploded mine on the clicked cell
- (5 points) Right click on a closed cell that contains no flag: Put a flag on the clicked cell
- (5 points) Right click on a closed cell that contains a flag: Remove the flag from the clicked cell
- (5 points) Left click on a closed cell that contains a flag: Do nothing
- (5 points) Change size and reset: Start over with new size and new number of mines (if applicable) and go to Part I

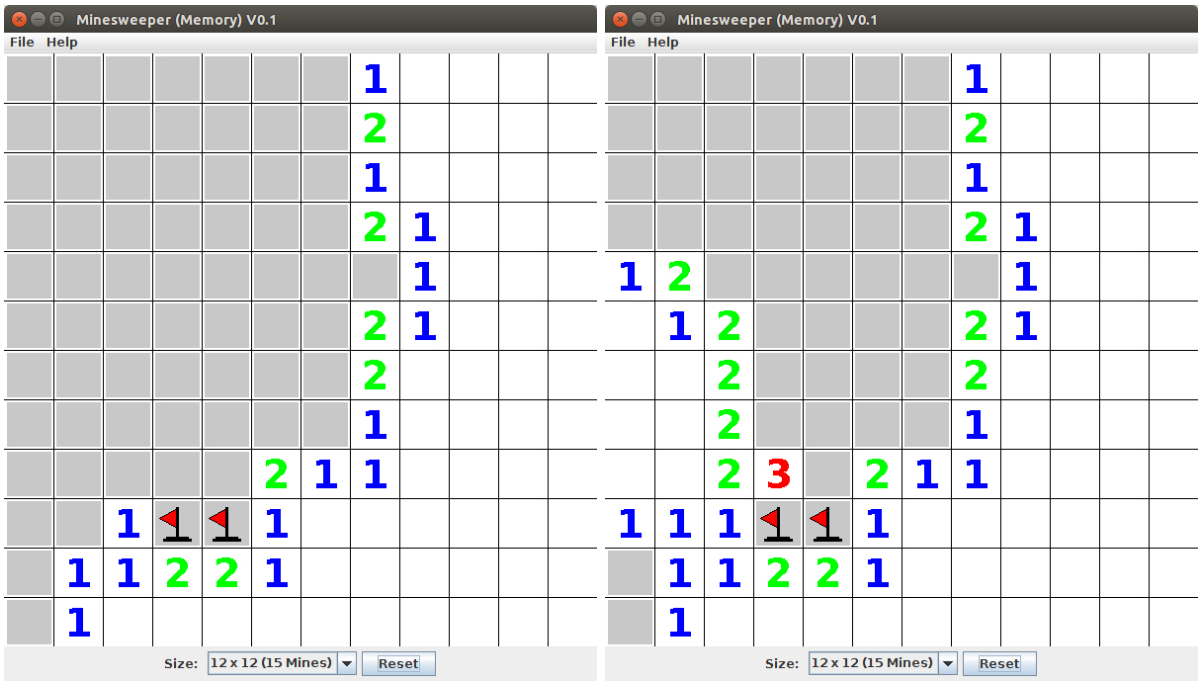
Part III: Recursively Open (30 Points)

For this part, we are going to perform a left-click on a close cell that contains nothing in it. Your program must perform recursively open all cells around the clicked cell as discussed earlier. Here is an example where the first image is a peek inside the game and the second image is the result of clicking on a blank closed cell:



Part IV: Left-Click on a Number (20 Points)

For this part, we are going to test your program by perform left-click on numbers. Recall that if the number of flags around a clicked number are not the same as the number itself, simply do nothing. However, if the number of flags are the same, you simply perform recursively open around it. Pictures below show before and after the left-click on the number 1 at row 9 column 2.



Part V: Status (10 Points)

For this part, your program should simply print the number of open cells as well as the number of flags on the Minesweeper hardware. **Note** that you have to keep track of them yourselves. Display the status every time the number of open cells or the number of flags is changed on the console screen. Lastly, if a user wins, just simply display "You Win!!!" on the console screen. Below is an example:

```
Number of Open Cells: 0 Number of Flags: 0
Number of Open Cells: 75 Number of Flags: 0
Number of Open Cells: 75 Number of Flags: 1
Number of Open Cells: 75 Number of Flags: 2
Number of Open Cells: 76 Number of Flags: 2
Number of Open Cells: 93 Number of Flags: 2
Number of Open Cells: 93 Number of Flags: 3
Number of Open Cells: 94 Number of Flags: 3
Number of Open Cells: 94 Number of Flags: 4
Number of Open Cells: 96 Number of Flags: 4
Number of Open Cells: 96 Number of Flags: 5
Number of Open Cells: 97 Number of Flags: 5
Number of Open Cells: 98 Number of Flags: 5
Number of Open Cells: 98 Number of Flags: 6
Number of Open Cells: 98 Number of Flags: 7
Number of Open Cells: 99 Number of Flags: 7
Number of Open Cells: 103 Number of Flags: 7
Number of Open Cells: 103 Number of Flags: 8
Number of Open Cells: 104 Number of Flags: 8
Number of Open Cells: 104 Number of Flags: 9
Number of Open Cells: 107 Number of Flags: 9
Number of Open Cells: 121 Number of Flags: 9
Number of Open Cells: 121 Number of Flags: 10
Number of Open Cells: 122 Number of Flags: 10
Number of Open Cells: 125 Number of Flags: 10
Number of Open Cells: 125 Number of Flags: 11
Number of Open Cells: 125 Number of Flags: 12
Number of Open Cells: 125 Number of Flags: 12
Number of Open Cells: 126 Number of Flags: 12
Number of Open Cells: 126 Number of Flags: 13
Number of Open Cells: 128 Number of Flags: 13
Number of Open Cells: 128 Number of Flags: 14
Number of Open Cells: 129 Number of Flags: 14
Number of Open Cells: 129 Number of Flags: 15
You Win!!!
```

Submission

The due date of this project is stated on the CourseWeb. Late submissions will not be accepted. You should submit the file `minesweeper_game.asm` via CourseWeb.