**Wentworth Institute of Technology**

**College of Engineering and Technology**

**Department of Computer Science and Networking**

**Assembly Language**

**Proposal and Requirements Document**

1. **Scope of the Development Project**

The project we have tackled is Minesweeper. The objective of the game is to clear a rectangular board containing hidden mines without detonating any of them, with help from clues about the number of neighboring mines in each field. Limitations include the Graphics and User input. Both are explained in detail within section 2. Distinct features include a modular grid, which will change as the value of a variable is changed and Color coded values within grid that will alert the user to the surrounding number of bombs.

1. **General Constraints, Assumptions, Dependencies, Guidelines**

* Graphics: Due to the time constraints of this project, we will be unable to provide a GUI for this project. We will compensate for this by using ASCII based visuals.
* Clicking: Due to the time constraints of this project, we will be unable to implement user mouse input. Due to this being such a crucial part of minesweeper, we will be implementing in its place a battleship-like X,Y grid system for the clicks. Along with plans for implementing mouse input if time remains at the competition of the project.

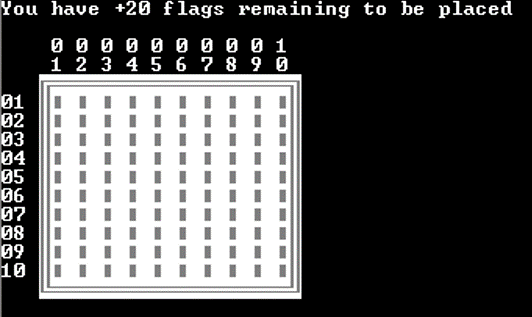
1. **Description of Functional Requirements**

***Generate Bombs***

* Purpose: To generate a set number of bomb’s locations, based off of the level
* Inputs: the procedure will take in a specified number of bombs (B) from a variable, along with the size of the grid.
* Processing: two random numbers between 1 and N will be generated and stored as X and Y. The values will be checked to make sure there is not already a bomb in the location.
* Outputs: The B number of values generated will be stored in parallel arrays, one for X and one for Y

***Printing the Grid***

* Purpose: This procedure will be used to print out a visual representation of the game field.
* Inputs: It takes in a variable N containing the grid’s size (N x N), and an array of chars that contains the values within the grid
* Processing: Within the procedure, calculations will be done for printing the correct number of values along the X and Y axis.
* Outputs: The proc will output an N x N labeled grid. Along with a message displaying the number of flags remaining to be placed



***Choosing what to do***

* Purpose: This is used to get what the user want to do: click on a location, flag a location, or remove a flag
* Inputs: The user inputs a char
* Processing: the char is checked to be a ‘C’ (click), ‘F’ (flag), or ‘R’ (remove flag), along with the lowercase variants. If there is a match, the a procedure is called
* Outputs: An error is outputted if the users input does not match the C, F, or R prompts. If the user attempts to flag a location and does not have any flags remaining, an error will be thrown, along with if the user attempt to remove a flag when none are placed.

***Clicking***

* Purpose: The purpose of this is to take a user’s X,Y value and manipulate the grid with it though either exploding a bomb, printing the number of bombs around, or clearing whitespace
* Inputs: The procedure will take in a user’s X,Y value.
* Processing: The procedure will double check the X,Y is within proper lower and upper bounds limits, then compare the X,Y value to the locations bombs have been placed, see if there are bombs surrounding the location, or it will clear whitespace
* Outputs: If there is a bomb in the location clicked, a variable will be set high. If there were bombs surrounding the location, the character representing the unclicked tiled will be replaced with the number of bombs surrounding the location. Finally if the user clicked on whitespace (not a bomb location, nor any bombs surrounding), the tiles representing unclicked space will be cleared out until there is no whitespace connected that is able to be removed (see photo below).



***Flagging***

* Purpose: The purpose of this is to take a user’s X,Y value and manipulate the grid with it, though placing an X in the specified location
* Inputs: The procedure will take in a user’s X,Y value and a variable containing the number of flags remaining
* Processing: The procedure will double check the X,Y is valid within the upper and lower bounds, that there are flags remaining to be placed, and that flags are not currently placed in that location.
* Outputs: The corresponding X,Y value in the grid will have the default character replaced with an ‘X’. Or an error is thrown if the values are out of bounds, a flag already exists, or there are no remaining flags to set

***Removing Flags***

* Purpose: The purpose of this is to take a user’s X,Y value and manipulate the grid with it, though removing an X in the specified location
* Inputs: The procedure will take in a user’s X,Y value and a variable containing the number of flags remaining
* Processing: The procedure will double check the X,Y is valid within the upper and lower bounds, that there are flags able to be removed, and that a flag is currently placed in that location.
* Outputs: The corresponding X,Y value in the grid will have the ‘X’ replaced with the default value. Or an error is thrown if the values are out of bounds, a flag doesn’t exist, or there are no remaining flags to remove.

***Updating***

* Purpose: This procedure will be run after the user selects what to do and before the grid is updated
* Inputs: This procedure takes in the bomb hit variable, and the number of flags remaining
* Processing: The code will check to see if the bomb variable is high. The code will also check to see if the flags remaining is equal to 0. If so, it will compare the flag locations to the bomb locations.
* Outputs: Should the bomb variable be high, the lose procedure will be called. If the flag locations match the bomb locations the level up procedure will be called. Otherwise the program will continue as usual

***Level Up/ Win***

* Purpose: To change the program to the next difficulty level
* Inputs: Takes in the total number of bombs, N (size of the grid), and level
* Processing: If level equals 4 (highest level), tell the user they won! Otherwise, five is added to N, the number of bombs and flags is updated, the level is increased by one. The grid is also cleared back to default.
* Outputs: A cleared grid of increased size

***Lose/ Replay***

* Purpose: To give the user the option to repeat a level or quit after losing
* Inputs: Takes in the total number of bombs, also prompt the user to see if they wish to continue.
* Processing: should the user not wish to continue, the program will quit. Otherwise, the flag number will be set to the total number of bombs and the grid will be reset.
* Outputs: A cleared grid of the same size.

***Checking the Range of X and Y***

* Purpose: To check the bounds of X and Y
* Inputs: takes in the variables X, Y, and N (size of grid)
* Processing: Checks to see that X and Y are greater than 0 and less than or equal to N
* Outputs: the procedure will store a 0 in dl if the X,Y values are valid or 1 if they are invalid

***Resetting the Grid***

* Purpose: To reset the grid to default values between levels
* Inputs: There are no inputs for this procedure
* Processing: the procedure will loop through the grid array and set each value to the default value
* Outputs: the array with the grid values has been reset to a “blank” state

***Check for Bomb***

* Purpose: To check and see if a bomb is in a user specified location
* Inputs: The procedure takes in X and Y, along with arrays for the locations of the bombs
* Processing: the X and Y values are compared to all the values of the bomb locations
* Outputs: If there is a bomb at X,Y a 1 is returned in dl, otherwise a 0 is returned

***Get number of Bombs Surrounding***

* Purpose: To get the number of bombs that are around an X,Y location
* Inputs: The procedure takes in the values X and Y
* Processing: The procedure will check the 8 values surrounding X, Y using the check for bomb procedure.
* Outputs: dl will contain the number of bombs surrounding the X, Y location

1. **Other Requirements**