Jeremiah Vannest
Arjun Pandya
LNC 10:20AM
APP_C39_1

MineSweeper Pseudo Code

1) The screen is cleared and the text minesweeper is displayed in the top center of the screen with green background.
2) A while loop will be created to make the game re-playable and after a few seconds, the screen is cleared and the entire background turned green. The menu is displayed with rules, easy game, hard game, statistics, and credits.
3) The selection will be done through the touch screen based on where you press on the screen.
4) If rules option is selected, the screen will be cleared and the rules will be printed. The screen will be pressed in order to move to the next page of the rules and back to the menu screen.
5) If Easy is selected, the screen will be cleared and an easy version of the game will start by calling the easy game function (described later) which is sent the pointers to the variable that stores the wins and losses.
6) If Hard is selected, the screen will be cleared and the hard version of the game will start by calling the function (described later) which is sent the pointers to the variable that stores the wins and losses.
7) If the statistics option is selected, the screen will be cleared and the number of wins and losses will be displayed. The wins and losses are tracked by the easy game and hard game functions. The screen will be pressed in order to return to the menu.
8) If the credits option is selected, the screen will be cleared and the credits for the game will write to the screen. The screen will be pressed to return to the menu.
9) The easy game function that is called in the main starts the easy game.
10) The 7x7 game board is drawn using DrawLine.
11) A 7x7 array called mfield is created that represents the graphical output on the screen and is filled with zeroes with nested for loops.
12) Srand is then seeded with the current time
13) A for loop is used to create 12 pairs of randomly generated row numbers and columns numbers that are used to fill 12 random locations in array mfield with ones which represents bombs.
14) A while loop with a check condition that will be changed if the player hits a bomb or wins. This makes the game continue running until the game has been completed.
15) If the screen is touched, the x position and y position that are returned by the Touch function will be converted to row numbers and column numbers that correspond to the 7x7 array mfield.
16) Then an if else statements are used to check if the that location in the array is equal to a one or a zero.
17) If the location in the array is equal to one, then that means the player has hit a bomb. The screen will be cleared and will turn red. “BOOM” will be written to the center of the screen and the buzzer will go off. The pointer to the loss variable will be used to increment the number of losses by one. After a few seconds, the screen will be cleared and the check condition will be changed to exit the while loop and return to the menu.
18) If the location in the array is equal to zero, then a series of if else statements are checked.

Jeremiah Vannest
Arjun Pandya
LNC 10:20AM
APP_C39_1

19) If the location in the array is the box in the center (everything not on the border of the array), then the 8 squares around that element in the array are checked for 1s and if there are bombs then the bomb counter is incremented.
20) The number of bombs around that square are then printed to the corresponding square on the screen.
21) If the location in the array is one of the corners, then the 3 squares around the corner location of the array are checked to see if they contain 1s and if they contain bombs then the bomb counter is incremented.
22) The number of bombs around that square are then printed to the corresponding square on the screen.
23) If the location in the array is on the top row, side row, or bottom row (not including the corners), then the squares around that location are checked for ones. If there are bombs, then the bomb counter is incremented.
24) The number of bombs around that square are then printed to the corresponding square on the screen.
25) The location in the array that was selected is then set equal to 2 (allowing it to be determined if the play won or not).
26) The elements of the array are then all summed and if the value is equal to 86, which is the value if all of the zeroes have been set equal to two and the 1s from the bomb remain, then the screen is cleared and "YOU WIN" is displayed in the center of the screen and the pointer to the wins variable is used to increment the wins by one. After a few seconds, the check condition for the while loop is changed so that the game returns to the main menu.
27) The hard game function called in the main function starts the hard game. The set up for this is the same as for the easy game just with different numbers as it is a 9x9 grid/array with 18 bombs.