Final project report

1. Breakdown of what I learn
    a. Using ncurses to create a snake game
        i. Creating the basic snake game
            1. Functions from ncurses library I learn when creating the basic game
                a. initscr()
                    i. Determines the terminal type and initializes all implementation data structures.
                    ii. Comments: I am not really sure what it means but I know that I have to call this function before starting using other functions in the ncurses library.
                b. clear()
                    i. clears the screen of the terminal
                c. noecho()
                    i. Do not show user's inputs in the terminal
                d. cbreak
                    i. Disables buffering of user's input so user input is directly available for program
                e. cur_set(0)
                    i. Setting the cursor to be invisible
                f. keypad(window, true)
                    i. Allows user to use the arrow keys in a given window
                g. refresh()
                    i. Printing output(changes made to standard screen) to the terminal
            2. Creating the snake tail
                a. Creating the snake tail was the hardest part of creating the game for me. Below is my understanding of how to create snake tail.
                    i. To create the snake tail, one has to first have the head of the snake. Head of the snake is first generated at the center of the playing area and the user will interact with the arrow keys to determine where the head of the snake will be.
                        1. Given the head of the snake, we want the snake to expand one "unit" of tail when it "eats" the fruit. Therefore, in the code we want to have a variable that keeps track of the number of units of tail that the snake currently has. In my code, I name this variable nTail which is initialized to zero. So whenever the snake head has the same coordinate as the fruit, I increment the tail by one.

        2. Next, we need to have two containers(array in my case) to store the coordinates of the tail. So how do we get the coordinate of the tail? The underlying trick is that the coordinate of the unit of tail right after the head is the coordinate of the head of the snake before the snake moves and the tail that is two units away from the head is two coordinate of the head before it is incremented twice and so on.
        3. So what I did was to always have a variable that denotes the current coordinate of the head of the snake, snakeX and snakeY for the x,y coordinate of the snake. These coordinates will always be stored as the first element of the array for the x-coordinate and the first element of array for the y-coordinates. If nTail is not incremented, the current x,y coordinate of the snake will always overwrite our previous x,y coordinate of the snake. This can be done using for loop with condition that we do not store any more coordinates after nTail.
        4. On the other hand, if nTail is incremented when the snake eats the fruit, the first element of the arrays will be the previous coordinate of the snake and the second element of the arrays will the coordinate of the head of the snake before the coordinate of the first element in the arrays. Therefore, we will first store the second elements of the arrays as prev2X and prev2Y and the first elements of the arrays as prevX and prevY. Then, we will store the second element in the arrays into prev2X and prev2Y and shift the values of the first elements in the arrays to the values of the second element in the arrays. Lastly, we modify prevX and prevY to store values of prev2X and prev2Y since we have moved some elements of the array to the right by one and iterate the whole process for the length of the tail -1 (we start with 0 index)
3. Additional stuff I add to the basic game
    a. Game over when the snake ‘eats’ itself
    b. Menu

            i. When the user compiles the program, they are greeted with a welcome page that asks them for the difficulty level of the game. There is a small window which has two choices for the difficulty of the game that the user can choose from, Normal and Hard
                1. Functions I learn when creating the menu
                    a. c_str()
                        i. converts a string into an array where each element of the array is a character of the string
                    b. WINDOW* window_name = newwin(starting y-coordinate, starting x-coordinate, y-coordinate length, x-coordinate length)
                        i. Creating a new window in the standard screen with starting at a given coordinate and have the given dimensions
                    c. wrefresh(window)
                        i. Printing the output(changes made to a particular window) to the terminal
    4. Where I learn how to create the game
        a. Basic game
            i. Three parts so three links
                1. https://www.youtube.com/watch?v=OAv2QsOZ4l4&list=WL&index=31&t=5s
                2. https://www.youtube.com/watch?v=fZIFwbYamk8&t=35s
                3. https://www.youtube.com/watch?v=MEjaEBv3rQ0
        b. Menu system
            i. https://www.youtube.com/watch?v=3YiPdibiQHA&list=WL&index=32&t=0s
ii. Other comments (IMPORTANT)
    1. I initially started watching a similar video for creating the basic snake game but the person was using conio.h a library that is only available for windows computer. Therefore I looked for the alternate library for MacOS on the internet and happen across ncurses. Fortunately, I was able to find a video on youtube of a person using ncurses to create a snake game.
    2. Also, this project is the first time I use a text editor and then compiling the code at the terminal. This is because Xcode, which I

            always used to program c++, does not seem to connect to the terminal directly.
        3. As I was learning to create the menu system for my game from a separate video, the person was using makefile to help him compile and run the code and recommended incorporating this method. Since professor also mention makefiles in class quite a few times, I decided to learn the basics of makefiles too.
    b. Learning how to use and create a makefile
        i. Creating a simple makefile
            1. Steps
                a. Create text file
                b. Have target, dependencies and action
                    i. E.g. main.o: main.cpp /n g++ -c main.cpp
        ii. Using a makefile
            1. make command
            2. call file
                a. ./output
        iii. Where I learn how to create a makefile
            1. https://www.youtube.com/watch?v=_r7i5X0rXJk&t=435s
2. Improvements I can make on the project
    a. Simple ideas that I could implement
        i. Having more bombs as the score of the user increases for Hard mode.
            1. Probably write a function that takes in the score of the user and that outputs a certain number of bombs for that score and then creating these bombs such that they do not have the same coordinates
    b. Other ideas that can improve the game
        i. I initially wanted to use Qt creator to create the game but find that too daunting so I decided to use ncurses library instead. Now with some experience in creating the snake game, I can better focus on learning the Qt framework to create the snake game