P#1 - MANAGERS REPORT

1 – HURDLES:

Some of the biggest challenges I encountered while completing this project had to do with the curses library. It wasn’t necessarily flaws in the library itself that was causing me trouble, but rather my own implementations of the curses functions. I spent the majority of my time fine tuning my abstract window toolkit classes than I did working on pointer power itself. Fortunately, the more I expanded my UI library, the easier it became to quickly finish a project. When it came to pointer power itself, my biggest hang up involved trying to figure out how to loop in order to fill my vector of vectors.

2 – PRIDE:

After this project is all said and done, the thing I’m most proud of is my UI library that’s built around PDCurses functions. I’ve taken to calling Curses Abstract Window Toolkit after Java’s AWT, because I tried to emulate some of the same functionality and class structure. Developing this library is an ongoing process, but it has helped me learn a ton about object oriented principles. My hope is that one day, others will find and use my library in their projects.

3 – HELPED OTHERS:

I sit right next to Nils in class and we often bounce ideas off each other and demo our projects to one another. I usually help him by suggesting visual alterations or addition features he could add to his program, as he’s a more advanced programmer than I am. Even so, I’m still able to sometimes assist him with problems he’s having with the curses library.

4 – HELPED BY OTHERS:

Nils has been an immensely valuable resource for me this semester. Since he has a stronger grasp of object oriented principles than I do, he’ll often look at my code and help me out when otherwise I’d be stuck. He also comes up with great ideas for features which I then go on to attempt to implement myself. Not to mention, he also introduced me to PDCurses, without which I would be completely lost.

5 – STARS:

**STAR: Prompt user for full (first and last) name. Burp back full name and separately display just the first name and just the second name.**

**STAR: After user signs in, use the user name in all future prompts, like:**

**STAR: Center-align the columns:**

**STARS (2): Add a “histogram” after row #9**

**STARS (2): Alternatively, do the previous STAR with the histogram rising “upward” instead of going “downward”**

**STAR: The value is the number of balls that will be “dropped”. Accept only numbers > 0 and < 10000 (or whatever number you select). And smallest value>0 to crash program is?.**

**STAR: Use vectors of pointers, not arrays of pointers, throughout this assignment**

**STAR: Perfectly align the Weight and SCORE values**

**STAR: Create two menu items, one to write the Game History to a disk file and another to read the Game History from the disk. When performing the disk writing and reading, be sure to “echo” to the screen the data being written and read.**

**STAR: Expand the previous STAR to give user the option to “append” data when disk writing.**

**STAR: Also display a “High Score” listing (using a sorting function you custom-create via any sorting algorithm):**

**STAR: Expand the previous STAR to include the corresponding Game #:**

**STAR: Expand the previous STAR by creating two menu items: one to write the High Score Table to a disk file and another to read the High Score Table from the disk. When performing the disk writing and reading, be sure to “echo” to the screen the data being written and read.**

**STARS (2): Allow multiple users to sign-in and keep separate High Score Tables for each user**

**STARS (2): Allow multiple users to sign-in. In the High Score Table include the user’s name with the high scores**

**STAR: Only proceed after sign-in is done, although it’s okay to select exit**

**STAR: Instead of re-writing the menu for “Sorry” message, be “ecological “and move the cursor back to the prompt point, but be sure to pause for a moment and then erase the now unnecessary “Sorry” part of the message**

**STAR: Expand previous STAR to be ecological for all appropriate menu selections**

STAR: Effectively implement a pointer to a pointer

STAR: Effectively implement a “smart” pointer

STAR: In a NEW engaging way, use >=4 windows.h colors throughout

STAR: Implement this assignment by creating and using a custom header file to hold all your function prototypes and definitions. Include your Michelangelo documentation with ID INFO, etc. in the header file

STAR: Effectively use three new "Advanced" features (= not yet introduced). List them:

Classes, inheritance, constructors

**TOTAL:**

**27**