Final Project Report – CSCE-155N

My final project for this class was a baseball stat calculator for batting statistics which was approved by learning assistant Thien Cao. I wanted to do this because baseball is one of my favorite past times and I wanted to create a program that would allow me to calculate those statistics that go right along with tracking the success of the batter. I only implemented the 3 most prevalent batting statistics but there are numerous more that could be added.

I began by just creating a small figure that is similar to that of the vending machine figure from Lab 11 that would act as my calculator. Next, I needed to figure out which inputs would be needed to get the proper statistics which for most of them would just be the amount of at bats, hits, and walks. But to calculate slugging I needed the number of singles, doubles, triples, and homeruns. So, the inputs that I decided on would be outs, singles, doubles, triples, homeruns, and walks. This would be enough information to calculate batting average, on base percentage, slugging, and on base percentage plus slugging which I then created buttons to calculate these stats the user. But then I also wanted to make my program a little more complex so not only would it be a calculator but a program that would also track the stats for a specific player. So, the inputs I then added was an input player number edit box and radio buttons for the user to select home or away. The user after inputting all the proper player information could then press the enter button at the button of the calculator which would display the players statistics in a dialog box but would also create a csv file with all the statistics just calculated. In operation my program could just be simply used as a calculator but with the addition of some other functions I have allowed the user to track the stats of their players or the other teams.

I began by creating my uicontrol buttons which included all the user inputs as well as buttons mentioned earlier to calculate the stats. I then created the displays for the output of that statistics that would be shown on the middle of the screen. I used the same basic concepts from Lab 11 to create the display for the output which would display the string of a certain value. These uicontrol buttons were present in my main function along with the initial values of each variable that I created for the inputs as well as the outputs. Each pushbutton/radio button then had a callback function attached to it with my program summing up to 15 different callback functions. I then created my callback functions that would go along with those uicontrols which each acted in their own accord to display the proper output. I had a callback function for each of the outs, singles, doubles, triples, homeruns, and walks that would either at bats, hits, walks or a combination of them updating those proper variables. Then in those functions there would also be proper display uicontrols to update the displays in the middle for the user.

Next, was the implementation of the calculate push buttons which would take in the variables that were just updated from the other push buttons and display the proper statistics in the middle of the screen. I began by looking at what variables would be needed for each stat and then I created a proper equation that would give the proper output. I had trouble with the code displaying a “NaN” message if all the inputs were still 0, but then I just created an if statement that would check if at bats were 0 and then keep the display 0 without running the rest of the callback function. I also originally did not have the single, double, triple, and homerun buttons adding at bats to the display so the user would have to add at bats manually. So, I fixed it by having those buttons add 1 to the at bats display and make sure everything was properly updated to involve less user input. Throughout the code there are still if statements that are unnecessary as they check if hits exceed at bats, then it will throw an error message but with that update the error message and if statement are just there in case the user somehow finds an impossible way to do it. Then the calculate uicontrol buttons act just like the input buttons take in the variables just inputted and output on the display the proper statistics. The reset and close button callback functions were easy to write it was just reassigning the baseball variables to 0 and displaying them on the screen as well as just a close figure function. From there it was just some more aspects other than the calculator, so I added the edit box to input the player number which I had to do some research on how to do that but once I found it, it was simple. I created the home and away radio buttons and found functions that would then press the home button off if the away button were pressed.

From there it was just making some of the code more efficient for the user such as creating an if statement for the player number edit box that would throw an error if a non-numeric character were present. I also in the main function added a line of code that would automatically have the team has home just to allow the user more ease if he is tracking for his own team. Then in the final days of creating the code I realized I could make it a lot easier for the user by calculating the batting average, OBP, SLG, and OPS for them if they did not want to use the calculate buttons. I called the functions to calculate those stats in the enter button and it would automatically calculate them without using those calculate buttons that would allow the user less button pushes if he is trying to stat track. After much bug testing I was satisfied with my user interface and checked numerous times for anything that might cause an error in calculations.

Using Git was seamless at first, I was kind of just using the steps from lab 11 to upload my progress and it was working fine. Until I had an error and hit a merge conflict that was not resolving even with the use of the instructions to resolve a merge conflict form lab 11. I had then began looking up functions and came along to a git command called revert and thought that would be what I would use to resolve my conflict. But I did not realize this was the exact opposite of what I wanted, and it reverted all my code back to the progress of the previous commit. This resulted in a lot of rewriting code that shouldn’t have been this hard, but I learned from the experience and learned what I should do instead. I then committed my work, and all my code was uploaded correctly. This overall was a fun project to code and besides rewriting it I think I learned a lot from this project.