**Developer List:**

Katerina Betts

[bettska@mail.uc.edu](mailto:bettska@mail.uc.edu)

**Connect Four Final Project**

**Description:**

In this project for EECE 1080/CS 1, the classic game of Connect Four was recreated within the parameters available in the basic Cloud9 IDE. While playing the game, the player can decide whether to play alone with artificial intelligence or together with another player. Connect Four starts with one player inputting either a red or a black piece into a 6x7 board and taking turns with the other player who is placing red pieces. The game is 2-Dimesional to win, and it is won when one player receives four of their color pieces, red or black, diagonally, horizontally or vertically. As the player continues to play, the amount of wins and losses is totaled after every win and/or loss right before asking if they would like to play yet another game. The players are also able to play the game until they desire to quit.

**Original Specification (Word for Word):**

“For the end of the semester project, I am planning to do a C++ coded Connect 4. In order to do this, I plan on finding files or pictures online of basic or typical component of a Connect 4 game. This would include the black and red pieces, the “board” and potentially some form of character for the player and the AI. Of course, this would involve utilization of pictures and outside files in C++, which I haven’t done, so I believe it would be in my best interests to read a little more on stackoverflow.com. or cplusplus.com in order to better understand how to implement this feature of the game. It may also be in my best interests to keep track of the number of wins and losses overall before a player quits the game; however, I do not believe that is done in a regular game of Connect 4.

In order to successfully implement AI, there needs to be a srand() and rand() within the class or structure that the AI is held within. Despite the AI maybe not being programmed intelligently, it will be able to make moves based on a random generator until that part of the program can be fixed at a later time when it is nearing completion. It is also important to make sure if someone were to pick a two-player game instead of one-player that they would be able to put the AI in the background as necessary.

Another important part of the game is to make it interactive for the user, or users. In this case, I believe the best option to do this is utilizing either numbers or letter for the columns of the “board.” In order to do this successfully, I believe assigning numbers in an order like 1 2 3 4 or A B C D would indicate which column the user would like to place the next piece. As the piece falls, I might try to see if I can do some sort of falling animation like Dr. Mario because I think those kinds of graphics give off the same kind of vibe my project is going to give off. Also, using perhaps a sort of theme may have to include something to drop the red and black pieces with into the board.

When it comes to classes, I plan on having a Game, User, and AI class with the potential of having a class dedicated to files. I don’t want to put too much “stuff” into one class in the beginning because it will cause clutter, but if at the end it is not useful to have so many classes, I might combine AI and User into one class and Game and Graphics into another class. There shouldn’t really be anything in the main other than calls to the game class because I want to just call the AI, User and Graphics classes within the Game class and its functions.

For my personal creation of the game, I may have to think more as I’m coding to see what’s happening. A visual aid before programming will definitely be constructed, and that shell should be able to help me ensure that the base parts of the program are covered. Commenting the code will also be a necessary step-by-step process instead of the commenting after for the more basic lab code.”

**Program Feature List:**

* Single and Multiplayer
* AI Player
* Gameboard Display
* Win Counter

**Testing Methodologies:**

As each different function was written, a different set of test codes were hard coded into the main just to ensure that the function worked properly given different variants of the possible inputs. For example, when creating the function that placed the pieces, there were a few values hard coded into the main to visually see the correct output. Another way the code was tested was by playing through the game and trying to put in certain inputs to “break” the game or to find errors. This was done in an attempt to make sure that everything that a user puts into the game will have some sort of consequence without having the game do something out of the ordinary.

**Program Usage Guide:**

1. To first begin the game, click the run button in the IDE that is being used.
2. Once prompted by the command window, input whether or not there are one or two players using 1 or 2.
3. A board will be displayed and player one will be prompted to place a piece. This player should choose the column in which the piece will be placed 1-7.
4. If the user chose to have two players, this is where the second player will have a prompt similar to the first player’s, and they will be able to place a piece on the board.
   1. If the player chose to play with AI, the AI will be able to take its turn instead of a second player.
5. Once either player one or player two win the game, the command window will output a prompt that says either player one or player two has officially won the game. Another statement will be output as well asking whether or not the player(s) wants to play another game.
6. If the player inputs a “y” when prompted, the game will go back to step 3 to start playing the game again.
7. If the player inputs a “n” when prompted, the game will end, and the wins and losses of each player as well as the number of games played will be displayed.

**Lessons Learned**

Although the program implemented follows the same basic programming concepts learned throughout the semester, there were lessons to learn in the process of creating the code. For example, when I first planned on creating this Connect Four game, I planned on creating some form graphical user interface, or GUI. This, however, turned out to be a lot larger of a task than I expected. When I searched through the internet on ways to implement this particular idea, it appeared that I would have to download an external dependency. This is an unrealistic expectation as not only would I have to download this external dependency, but the professor would have to do so as well. This would not be a problem, but finding the correct one to download along with learning the GUI interface and implementing the difference features. This would be possible given I had allotted more time to work with the interface. I put too much time into creating the C++ portion of the program without the interface and did not suspect that the implementation would have taken as long as I learned it would.

Another lesson I learned from programming this game is that the win checking is quite hard for Connect Four. The most difficult part of implementing this function in general was the diagonal win. Due to the versatile nature of the diagonal wins, there are two different if statements to do the down left win and the down right win. This is because the diagonals can go both left and right, and it was a little bit harder to try to implement it all in one single if statement with a for loop. Of course, this is more of a small lesson in array matrices and how to utilize them properly. This may be because both dimensions of the matrix need to be iterated at the same time in order to check in a diagonal direction. It also could be a lesson in how an idea is easy in theory, but it is a little bit harder in practice to implement.

I also mentioned in the original description that I would make a class for the AI and the player, but by the end of the program, I had realized it was better to just make the AI and player inside of the class mostly contained in a properly named function. I also used counts for the player(s) and AI that that store the wins and losses as the game is played. This was originally going to be put into a different function in the program. Instead of putting the count variables in the player or AI function, I decided to put the count function within the play function. These may not seem like a lesson as much as each seems like a minor statement, but it was a lesson for me. Sometimes I try to hard to put certain features into my program that will not work the way that I envision them, which might be a good way to work around ways to implement a solution, but it’s also a way to get stuck on parts of code that are not as hard as they need to be. Implementing a basic AI in a function was a more practical thought for a program like Connect Four, but I was thinking more of a bigger, reusable picture with games in general. Sometimes it is okay to just look at a smaller picture in order to present a more efficient method of outputting a game or program.