4 December2017

**Developer List**

Tess Newkold – newkoltj@mail.uc.edu

Zahra Briggs - [briggszk@mail.uc.edu](mailto:briggszk@mail.uc.edu)

**Project Title**

Hangman!

**Final Project Description**

When the user decides to play our game a random number generator, generates a number between 1-10 each number corresponding to a predetermined word. A hint is given to tell the user that the words are all associated with finals week. The words are; finals, classes, coffee, stress, caffeine, programming, sleep, calculus, soda, and sandwiches. The game prompts the user to pick letters of the alphabet to try and guess the word. If the letter is contained in the word/phrase, the user takes another turn guessing a letter. If the letter is not contained in the word/phrase, a portion of the hangman is added, including the head, arms, body, etc. If the word/phrase is guessed (all letters are revealed) the user wins, if all the parts of the hangman are displayed the user loses. The game keeps track of the letters guessed, so if you guess the letter again it will not penalize you for guessing the wrong letter twice. The game also keeps track of how many guesses you have guessed wrong, because you only get 5 wrong guesses before the entire man is displayed. Finally, it tells you how many guesses you have made, because you can guess more than 5 times if you keep guessing correctly.

**Original Project Description**

The user selects a letter of the alphabet. If the letter is contained in the word/phrase, the user takes another turn guessing a letter. If the letter is not contained in the word/phrase, a portion of the hangman is added, including the head, arms, body, etc. If the word/phrase is guessed (all letters are revealed) the user wins, if all the parts of the hangman are displayed the user loses.

**Program Feature List**

Our program is not multiplayer. It is the user playing the computer. The computer does not really have an AI component, the computer is the administrator of the game. The computer does however, generate a random number to randomly select a random word out of 10 predetermined words.

**Testing Methodologies**

We tested the project by writing the individual functions separately and compiling the code over and over again until they worked properly. We then put all the functions together in a class structure and ran that over and over until we made 100% sure that the program compiled correctly, the way we wanted it to, and without errors or warnings each time.

**Program Usage Guidelines**

The usage guidelines are very simple. When you run the code the game automatically starts as we assume you initially want to play if you pressed run. The computer randomly generates a word for the user to guess letter by letter, and then leads the user through what to do. It prompts the user to guess a letter and tells them if they are wright or wrong and displays the letter in the correct order, or displays another body part, respectively.

Once the user has correctly guessed the entire word, or runs out of guesses and the man has been hung, the computer asks if the user wants to play again by pressing either ‘y’ or ‘n’. ‘Y’ will play the game again and ‘n’ will exit the code. The game will keep playing until the user hits ‘n’ when prompted.

**Tess – Lessons Learned**

I learned many things from writing this project. The biggest thing was how hard it is to read what other people have written. I anticipated this, because I think my brain works much different than others, but it didn’t end up being as difficult as I thought it was going to be. it was still a little weird to see other people’s code in something you are doing and try to make it a cohesive program with a similar style.

I learned how to use nested loops. I thought I knew how to do them, but they were really tripping me up and causing the game to be really messed up. Ade helped me understand how to get them to work, and I had to re-read the zy-books many times and google how other people did it too. I really struggled with getting these to work. I learned that it is very hard to fix code when you are tired. Nothing works when I am tired I guess.

I learned how to share code on Cloud9 with other developers. We could see each other working on the code in real time which really helped understand the other persons thought process in the code they wrote. It was also cool because you could see who contributed to which part of the code, because it became color coded with each person’s type. However, in the end it all ended up being one color because I copied and pasted all code into a new final draft to make sure it was all perfect and saved in the correct document.

I learned that starting early really does pay off. We may not have finished the program until late last night and early today, but we started at the beginning of November thinking about it and writing it out. This allowed me a lot of time to think about how I wanted to do write my part of the code and then plenty of time to actually execute the writing of it. I usually wait until this last minute to get things done. This time we really planned things out and it was much less stressful.

Lastly, the biggest thing I learned is that the program is not always as straight forward as the coder thinks. I thought the code we wrote was very straight forward and then I decided to play the game with my mom, and she was kind of confused about what to do. So, I changed a few things, especially spacing, to make it more user friendly and readable. I was surprised by this, but it makes sense that someone who has never seen the program before isn’t going to immediately understand how to play. The developer has been writing it and compiling the code so much that they are used to every part of it, and a new user has no idea at the start how things work. We tried to make it more user friendly and I played again with my mom and things went much better, but who knows, that may have been because it was the second time she played. However, I learned a lot from this experience, and I am very glad I took this class. I can’t believe how much I have learned these last few months which was emphasized while writing this final code project.

**Zahra – Lessons Learned**

This was a very interesting project to work on. I learned a lot of little things along the way but I think my over-arching lesson learned would be that programming is truly a process. Everyone has a process unique to their work ethic but the most important thing is recognizing that you have to have a roadmap in place in order to reach your final destination. In my case, I knew that we had to start the project in writing. After we submitted the draft of our hangman game, we sat down and thought out the general components and functions we would have in our program. This was my starting point, and trying to go straight from this to typing in code was a real challenge. I had to take some time to refer back to the book and do a bit more research on how things would work before I could start drafting up anything for my actual code.

I also had to write things out on paper a lot, which was another lesson I learned. It was easier for me to follow things and comprehend what was going on in the program when I had a skeleton on paper. In doing the labs during the semester we were always given a skeleton, so it was interesting to create the code that I created from scratch, while figuring out all the different ways to get what I wanted to get accomplished.

There were a few times during the actual testing and revising process when I would get something working, and then try to make it better but would end up messing everything up. There were also times when I went through several changes from the original code I had written, just to end up with the same results. During this time I wished I would have made multiple copies of my code that was working at that instant. That way, it would be easier for me to go back to an older working version if I had messed everything up. Ctrl+Z saved me many times but if I had just made copies of the code I could have saved a lot of time and frustration.