Project 3 Report - Anna Glass

To prepare for the project, my partner and I did individual research so we could have diverse ideas going into our brainstorm session. We read through examples of past projects on Moodle, looked at projects online, and came up with a list of movies, video games, and/or books that could inspire an idea for our project. We then zoomed to talk about our ideas and came up with a few specific ideas we wanted to work towards and examples to blend together for our project outline and code skeleton. We identified the characters, an end goal, a storyline, and a point system to put down the project framework we would build on.

To develop our code skeleton, we first came up with our class names: Challenge, Animal, and Menus. We wrote our header files, discussing first what private variables we would need (the animal stats, user choice, etc) and then what member functions we would need to manipulate those variables, what kinds of menus we would need, and what functions could serve as our challenges. Once we identified these functions, we talked about the basic steps that would go into each of these functions to give the desired outcome. We went mainly with a step by step approach, numbering steps that the function would take while keeping the project requirements in mind. When we started developing our pseudocode into actual code from our code skeleton, we essentially turned the outlined steps into comments, then did the code that would complete each step under those comments. This was really helpful to keep us on track and keep the main goal of each function in mind as we went, which prevented any confusion in remembering why we made those functions in the first place.

I think the main flaw in our process was not testing the code as we went. My partner and I couldn't get together and work on the same computer to test, and there really was no easy way to edit the same code window at the same time (in GitHub we had to commit our changes, reload the page, etc.), so it was difficult to test our code as we went along as testing involved one of us sharing our screen on Zoom and making the edits while the other could only watch and offer solutions to the problem at hand. We decided to just write the entire project, then once we were finished, create a main and debug at the very end. This definitely added time and wasn't very efficient as we had to debug absolutely everything that was wrong, instead of working through each function at a time. We could have researched a bit more to figure out a way to work on the same document real time, and test as we went along, or delegate debugging between us along the way so by the end debugging wasn't a big problem.

A really impactful false start we had was our Challenge class. We created it with a bit of confusion as we didn't know at the time the most efficient way to take the user through the challenges and different levels, and we didn't entirely know how to make animal objects and use only the object the user chose for the challenges. We had these functions in our code skeleton that were really daunting and confusing, involving a matrix, a bunch of boolean functions, among other things to compensate for the fact that we didn't know how to actually make objects for our project. In our design meeting, Abhilash mentioned how we needed to create objects for our animals, setting us on the right path away from these confusing challenges functions. In the end we actually ended up deleting the challenges class, which led us to completely change how we thought about the challenges and levels.

Originally we wanted to make different levels within the challenges based on which animal the user picked, but realized this would have been an insane amount of storyline planning and would become less of a coding project and more like a creative novel. So, instead of doing

this we limited it to a series of “mini-games” within the challenges that the user goes through no matter what. Once all of their animal’s stats reach 5, we take them to the “ultimate challenge.” This gave a lot of focus and purpose to our project, and allowed us to focus more on the actual code rather than creating an unnecessarily intricate story. And once we created the Play class for all of the challenges, we realized the Challenge class was entirely unnecessary. It also allowed us to fully understand why we needed animal objects and how to use them in our Play class.

This false start taught me that when I hit a spot of confusion early on, even in the planning stage, I need to face that problem and confusion head-on instead of finding ways around it. Had we addressed the confusion in the beginning before we wrote the Challenge class, we would have never written it in the first place and had our animal objects to work with from the beginning. Also, this would have given us the extra time and focus to really create a solid storyline that makes sense and makes for a really fun game. Overall, this project taught me how much effort actually goes into even the most simple games, and how planning, research, and understanding from the very beginning is really important for efficiency and focus through to the end.

I chose to work with a partner on this project as I have never worked with anyone with coding, and thought it would be beneficial to go through the process of creating a program with another person to experience the diversity of ideas that comes from multiple people working towards a common goal. Collaborating on this project was really eye-opening and created computer science as a collaborative discipline for me. At the beginning of the semester I didn’t understand the importance of collaborating with others and seeking help from mentors, but this project showed me that seeking out new and different approaches for coding always improves the process of programming and creates a better product.