The program that I have created is meant to tackle the idea and concept of process creation. In order to take advantage of this concept, I have created a program that will calculate Collatz Conjecture starting at a given n. Then we will create two child processes that will calculate two different numbers. First, the first child will calculate the input from the user (known as n) and the second child will calculate the input and add four to it. The hope is that the program will return a sequence of numbers that will eventually end at one.

Before I tackle the problem of creating two children from the parent, I wanted to at least get a function that will calculate Collatz Conjecture. Since the idea of Collatz is to check if the number that is being presented is either even or odd and then calculate it, I had to find a simple and effective way to loop through the sequence. Recursive function was a good pick to loop through the sequence without a need of a for loop. This will also allow me to call the function again for the other child. The rest of the program is simpler to explain. First we get the input from the user from the terminal, next we create a buffer and the child process from the parent. I make sure that when I forked, that the parent is the only one who creates the second process. Next we store the input into the buffer and finally print it out.

Looking at the results, there is some interesting finding that the program presents. First, to prevent the process from ending short, we call the wait function twice to make sure that the child finishes running the function. Next, my program runs the first child first and then the second. Using the example as a reference, I see that another potential output is the second child to process first and the first child to process second. Personally, I think that the way that you logically walk through the process will determine how the output will show. Finally, experimenting with printf and comparing the sprintf shows how the process will overlap when not properly stored in a buffer.

Overall, creating a child process helps to show how we are simultaneously solving problems with the use of a fork.