Project 1 Midpoint Report
Andrew Bare (aob1) – Dante Gordon (dxg2)
September 3 2023

## Project Description

The purpose of this project is to create a shell executable that can run on Unix like systems. In part 2 the program will be able to execute other programs and display the output in the shell. To facilitate this in part 1 it was required to accept and tokenize user input. Part 1 also required recognition of input redirection, output redirection, and backgrounding. In part 2 tasks will run in the background while new commands are input. Also, in part 2 data will be redirected from another input source or another output source, so it is important for the shell to recognize these properties.

## Achievements

In part one of the project, we created a functional main and a class called param that stores input commands from the terminal into a dynamically sized array of strings. We used the <string> and <cstring.h> header files for the getline() and tokenize() function calls. These functions were required to get the user input and then for splitting the input string on spaces and newlines. One issue we ran into while developing was not realizing that space on the heap needed to be allocated to store the tokenized input strings. Once space was allocated with new, the beginning functionality was easy to implement.

## Preliminary Testing

Testing was done with input strings in the terminal such as “Hello world” or “ls -la”. More in depth testing for working with redirects and backgrounding child processes was done with “<four”,”>five” and “&”. The program successfully parsed the tokens as well as retaining each status when tested, printing out errors when “<”, or “>” were input without a filename.

## Next Steps

Creating new processes will be done with fork()  these child processes open the door for background processing. It is important to take the proper steps to avoid zombie processes while working on part 2. Input and output redirection can be achieved using <fstream> to open input files and create output files, this in conjunction with proper logic will complete the input and output redirection requirements of part 2. To run processes in the background a combination of fork() to create child processes and exec() to load the correct program will be utilized. This will allow for processes to run in the background while new input is provided in a continuous manner.