Parsing was done by taking the string and taking that string character by character parsing it as it goes. Whenever the parser sees a quote, it sets the quoted flag and then doesn't turn off the quoted flag until the next quote appears. When a special character is seen (<, >, or | ), it is separated from the rest of the commands. At the end of the parsing, there is a vector of strings that is returned that contains all of the tokens that can be extracted from the string that is passed to standard input. Piping is done by forming a pipe using a set of two file descriptors. The standard output is replaced by the write end of the pipe and eventually the read end of the pipe overwrites the standard input. If there are multiple pipes, the input and output will be piped together by the standard output being used of one being used as the standard input for the other through the pipe. On the last command, the standard output is restored and the command is executed so that the result of the operation is displayed. Redirection is done in a similar way. Because output redirection can only occur at the end, I did not worry about the output redirection being piped into anything. For input redirection, I read the file in through a file descriptor and then I overwrote standard output with the write end of the pipe. Thus, if the input is piped, that input can then be used as input to other commands like the piping mentioned above; however, if the redirect is not piped, the output will just print to the standard output.