1. **Where do I look to get some guidance on getting started?**
   A 'where to get started' video has been put together for this purpose. Link to the Video (also posted on PA2 assignment): https://www.youtube.com/watch?v=YToA6gxWkVs

2. **My program rapidly prints out the prompt a bunch of times. What could be the reason for this?**
   This likely means your stdin is busted[1] and so reading input fails immediately and loops back around. The scenario this happens the most is when you're working on pipes -- you overwrite the 0 fd of the parent process and don't copy stdin back afterwards. You can fix this with dup and dup2.

3. **What should be the user name we use for the user prompt? Is it the user who is running the shell or the GitHub username?**
   It is the environment username - the user who is running the shell. You would resolve $USER using the appropriate utilities in C++ related to env. The function call, `getenv("USER")` will help you here.

4. **Does grep output anything in "grep /init"? For me it doesn't, whether I try running in the actual terminal or in my terminal.**
   Try something else other than *init*, for some reason in some machines *init* process does not show.

5. **Where am I supposed to connect "cd /home/" to? I am running on MAC OS, so I don't necessarily have a home directory.**
   /home is an absolute path to the home directory on linux; **you should only be running it in a linux environment, you cannot run your code on MAC OS.**

6. **How can I verify that I'm handling the background process correctly?**
   In the case of background processes that sleep (for example, `sleep 20 &`), use the command ps to check if the process shows up and then run ps again after the allotted sleep time is over to check if it finished.

7. **Is it possible for a child process to change the working directory of the parent process? If not, how should I approach the cd command? I'm able to change the working directory but only in the child process, so the parent's working directory never changes.**
   Don't fork to change directory. Just call the appropriate function from the parent then continue to the next iteration of the while loop. cd doesn't require an exec call.

8. **(Bonus - Sign Expansion) Anyone have any advice on performing the $() expansion? I am currently parsing out the inside and using my fork() loop to take care of the inside but I don't know how to get the last output of what's inside the $() into the outer command.**

One potential way to do it is to redirect the stdout fd to the write end of a pipe; your pipe implementation code will output to the write end of the pipe which can be read from the other end (read end) into a string using the `read(...)` system call. Once you receive this string, you can execute the whole command (by replacing the original sign expansion in the input with the output returned by it).

9. **It seems that after forking, my pipe has to be closed on both processes. Is there a mistake? Also, I don't see how to close the pipe after transferring control to the child process executing the command.**
   The pipe does need to be closed on both sides. So if you are redirecting stdout into the write end of the pipe, you should close the read side and vice versa for stdin. The side that you redirect stdin/stdout onto will be closed once the process whose I/O you redirected ends execution, since it'll treat the pipe like it would its stdin or stdout and perform appropriate clean up.

10. **Why are we constrained to `execvp(...)` to run the command as a child process? Why can't we use `execlp(...)` as well?**
    execlp is a variadic function, whereas the args are in an array of unknown length (at compile time) - there's no (good) way to flatten an array into a variadic function's argument slots.

References

1. “Your stdin is busted” - Teddy Heinen, TA FA21