The purpose of this project is provide experience with process manipulation in Unix.

**Problem Statement**
Complete the shell project described on pages 121-125 in the textbook. As explained in the text, this project is organized into two parts:

1. Creating a child process that executes the given command that the parent may or may not wait for
2. Implementing a history feature

**Assignment**

(20 points) Implementation. This project is to be done individually. The C code described in the text is available in the directory `/home/hwang/cs470/project1`. However, this project may be written in any language as long as it runs on csserver. Provide a makefile that will make your project if it needs to be compiled.

(10 points) Provide a high-level functional analysis and design of the program describing the functionality of the major components of the program and how they interact with each other, and a more detailed analysis and design for the data structures and algorithms used in the history feature portion of the program. If the program does not meet all of the project requirements, describe which parts are missing and what you think should be done to implement them.

(10 points) Provide a test plan for your project. i.e., give a list of commands that will demonstrate the features of your shell. Annotate this list with notes regarding what result is expected. The grade for this portion of the project will depend on how thorough the test plan is. Note that the test plan should cover all of the project requirements whether or not the program actually implements them.

In addition, answer the following questions:

1. What aspect of process manipulation did you find most difficult to understand?
2. What aspect of process manipulation did you find least difficult to understand?
3. What, if anything, would you change in your current design?
4. What, if anything, did you find interesting or surprising about process manipulation that you did not know before doing this project?
Notes on the textbook code
Here are a few notes regarding the C code from the textbook.

The `setup` function creates the `args` array by manipulating `inputBuffer`. It replaces all whitespace and any `&` in `inputBuffer` by the null terminator. The elements of `args` are pointers to the first letter of each word in `inputBuffer`.

After a signal handler is called, control goes back to the location of the interruption. This is usually someplace in the middle of the `read` function call, which causes the function call fail and return a negative value. Ideally, the shell program should loop back to reissue the `read` call in this case so that a user could ask for the history as many times as they wish. However, for this assignment you may assume that a user will ask for the history list only once between issuing commands.

What to Submit
Submit items no later than 4:30pm on the due date.

Create a tarfile containing the source code and makefile for the project. Submit this tarfile electronically by emailing it as an attachment to the instructor.

Submit the following items in hardcopy:

- A printout of well-documented code for your shell, preferably 2-up in landscape mode.
- The functional analysis and design of your project
- The test plan for your project and the answers to the questions above