For this assignment, we will use Perl to implement Harding's own shell, named **hush**. Figures 10-4 & 10-7 in the textbook describe how Unix shells accept input from a command line and then use the `fork()` function to spawn a child process which, in turn, executes the requested command. While the child process is executing the requested command, the parent process (that is, the one running the shell) should call a function, named `wait()`, so that the parent waits until the child is finished.

Fortunately, Perl supports the necessary functions. In fact, the following Perl code is an implementation of Figure 10-7:

```
#!/usr/bin/perl –w
use strict;
my ($command, $inLine, $pid);

do
{
   print "SomePrompt: ";
   $inLine = <STDIN>;
   chomp($inLine);
   # Insert code here to extract the 1st word of $inLine and store it in $command
   unless ( lc($command) eq "exit" )
   {
       executeLinuxCommand($inLine);
    }
} while ( lc($command) ne "exit" );



sub executeLinuxCommand {
   my $commandLine = shift(@_);
   $pid = fork();
   if ($pid < 0)
   {
      print "Unable to use fork() function.\n";
      exit 0;
   }
   elsif ($pid > 0)
   { # parent process will execute this branch of the if statement:
      wait();
   }
   else
   { # child process will execute this branch:
     exec($commandLine);
     exit 0;     # Make absolutely SURE that child doesn't get past this point!
   }
}
```

To the basic implementation listed above, this assignment requires that you add the following extra features:

1. Set the default prompt to: "`[hush:nnnn]$ `", where `nnnn` is the pid number of the current process that is running the shell script. For instance, the prompt might look like: `[hush:8667]$` if the current process had pid number = 8667. The pid number is available in a Perl program via the special variable named, `$$`

2. Add code that checks for the existence of a file named, `.hush_profile`, located in your `HOME` directory. (The leading period in the filename indicates that this is a "hidden" file.) Your Perl program will need to extract the name of your `HOME` directory by referencing the system environment variable, `HOME`, stored in the hash named, `%ENV`: `$home_dir = $ENV{"HOME"};`

   If this file exists and is readable, your Perl program should open it up and read the contents. The `.hush_profile` file should contain `<key,value>` pairs. These pairs should be stored in the file using one of the following formats:

   ```
   alias key = value     or
   set   key = value
   ```

   For example, a sample file might contain the lines:

   ```
       alias cls = clear
       alias copy = cp
       alias dir = ls --color
       set   PROMPT = Yes, Master?
   ```

These <key,value> pairs are used to define some aliases and/or "set" environment variables.

An alias allows one to customize the hush shell so that the user may use alternative spellings for Unix commands. For example, given the three aliases defined above, the user should be able to enter the following commands shown in bold text:

```
[hush:8667]$ cls                    (This command should clear the monitor.)
[hush:8667]$ copy file1 file2       (This command should copy file1 to file2.)
[hush:8667]$ dir                    (List files in current directory using color coding.)
```

A set pair is used to define an environment variable to be used by hush. In the previous example, 'PROMPT' is a hush environment variable that is given the value, "Yes Master?".

If the .hush_profile file doesn't exist in the HOME directory, your Perl program should continue without indicating an error. Of course, no user-defined aliases or hush environment variables will be defined.

3. If the .hush_profile file contains a value for a hush environment variable named, PROMPT, your Perl program should use its value for the prompt – rather than the [hush:nnnn]$ default.

4. The function: exec($command); will not work correctly if the command entered is "cd", because there is no actual Unix program named "cd". This is a built-in shell command for bash (and other shells). Built-in commands are not implemented by using the fork() function. Consequently, our hush shell will also need to implement this command as a built-in command in the context of the parent process. To accomplish this, your program will need to intercept the "cd" command and call the Perl function: chdir() rather than spawn a child process and calling the exec() function. **NOTE:** Given the command "cd" your program should *NOT* execute the fork() function.

5. There are two other commands that need to be implemented as hush built-in commands: alias and set. If the user enters the command, "alias", on the command line, your script should display in alphabetical order all of the currently defined hush aliases. Likewise, if they enter the command, "set", your script should display in alphabetical order the currently defined hush environment variables.

6. Whether or not the .hush_profile file exists, your script should automatically add two hush environment variables, CWD and LWD, that keep track of the current and last working directories. These values should change when the "cd" command is used to change the current working directory. In addition, add another built-in command, named last, that changes the current working directory to the value stored in the LWD environment variable (and, of course, changes CWD and LWD accordingly). (To implement this feature you will need to use the Perl function: getcwd(). This function returns the name of the current working directory; and requires that you include the command: use Cwd; near the top of your program.)

7. Append every command that a user enters while using the hush shell to the hidden file .hush_history located in the user's HOME directory. The contents of this file should accumulate across multiple uses of the hush shell. In addition, add a built-in command, named history, that will display the entire contents of the .hush_history file.

Summary of requirements:

- Define default prompt
- Read and parse .hush_profile
- Use environment variable, PROMPT, for the prompt rather than the default value
- Implement built-in commands: cd, alias, set, last and history
- Define and maintain built-in environment variables: CWD and LWD

Sample run:

```
[baber@taz test]$ hush
[hush:17763]$ pwd
/home/baber/test
[hush:17763]$ dir
hush  lset  scorelist
[hush:17763]$ copy lset lset2
[hush:17763]$ dir
hush  lset  lset2  scorelist
[hush:17763]$ history
1  pwd
2  dir
```

```
3  copy lset lset2
4  dir
5  history

[hush:17763]$ cd
[hush:17763]$ pwd
/home/baber
[hush:17763]$ alias
Alias: cls = clear
Alias: copy = cp
Alias: dir = ls –color

[hush:17763]$ set
Set: CWD = /home/baber
Set: HISTFILE = /home/baber/.hush_history
Set: HOME = /home/baber
Set: LWD = /home/baber/test

[hush:17763]$ last
[hush:17763]$ pwd
/home/baber/test
[hush:17763]$ set
Set: CWD = /home/baber/test
Set: HISTFILE = /home/baber/.hush_history
Set: HOME = /home/baber
Set: LWD = /home/baber

[hush:17763]$ exit
```