For this assignment, you will be debugging and extending an implementation of a simple shell program
that we’ll call “rubbish” (note that, like all good shells, it ends in sh).

The latest version of rubbish has been put in a CVS repository for you on the department Unix machines.
You should have a new repository in ~/rep, and there should already be a project named rubbish in this
repository. Of course, you don’t want to work with files in the repository directly. Instead, check out a copy
of the project in another part of your account, make your changes there, and then commit them back to
the repository. When the assignment is due, I’ll collect the final state of your project from your repository.
You won’t need to turn anything in or bring a printout. However, you will want to make sure you have
committed all your changes before class on the due date. Since this may be the first time you’ve used CVS,
I suggest checking out another copy of your project (elsewhere in your account) just to check and make sure
you’ve committed all the changes you want.

Makefile

In its current form, rubbish doesn’t have a makefile. I want you to create one like the ones we’ve been using
in class. Let’s have the makefile know how to build an executable named rubbish, let’s have it know about all
the dependencies so it can just recompile and re-link where it needs to, and let’s give the makefile a “clean”
rule so that it can delete all objects and the target executable whenever we want to make a fresh start.

To make our makefile a little more interesting, let’s also combine the expand and builtin components into
a static library, librubEB.a. This is a little silly for such a small project, but it is representative of how we
might organize a larger project, especially if the functionality in this library was needed by more than one
executable.

The rubbish shell uses the gnu readline library to read input lines from the user. You will need to link
with libreadline.a and with your own library to build the executable.

Debugging

In its current form, rubbish is kind of buggy. I’ve deliberately introduced multiple bugs into the code to give
you a chance to work with debugging tools. When you start compiling and running rubbish, it will probably


If a word contains a $ followed by a sequence of alphanumeric characters, then the sequence should expand to the value of the named environment variable. If the variable is undefined, you should just replace the expansion syntax with an empty string.

**Built-In Commands**

In rubbish, the runBuiltIn() function inspects the command line to see if it matches a built-in command. Right now, our shell doesn’t know any built-in commands, but you are going to fix that. You should implement a function in the builtin component for each of the following commands. If runBuiltIn() sees one of these commands, it should call the appropriate function to execute it and then return true. Your built-in commands should print out short, meaningful error messages if they can’t perform their function.

- **exit**
  Like the shell exit command, your exit will built-in will take one numeric argument. It will exit the shell with this argument as the exit status.

- **cd**
  Your cd built-in will take a pathname as the argument. It will change the process’ current directory to the given pathname.

- **pwd**
  The pwd built-in will take no arguments. It will just print out the name of the current directory.

**Output**

Your exit and cd built-in commands should not produce any output. If the command isn’t a built-in, just let rubbish print out all the expanded words on the command line.