(85 points) Start a new file for built-in commands, "builtin.c", and implement built-ins as follows:

1) Build a framework for executing built-in commands. These commands are done directly by the shell and are not forked to another process to do. Your processline() function should call arg_parse() and then call a routine to check to see if the command is a built-in command. This function may execute the built-in if it is a built-in before returning to processline(). You may also have two functions, one to check if the command is a built-in and one to execute it. Add the prototype definitions of the one or two functions to builtin.h. These functions should accept both the argument vector and the number of arguments since arg_parse() is giving processline() that information. After processline() determines it is not a built-in command, that is the time to call fork(). If you execute a built-in command, no fork() is called. Remember, your "shell" process must free the argument list after it is done using it, regardless of whether it uses fork() or not. Note: in later assignments you will be required to add more built-in commands. A good "framework" will provide a very easy way to add new built-in commands. While a series of "if (str-cmp(...,...)...) ... else if (strcmp(...,...)...) ..." will work, try to think of a way to use strcmp only once (in a loop) to determine which built-in command needs to be executed. (Function pointers are really nice to use here with having each built-in command in its own function. Look at the labs using funcptr and how to using funcptr with array of pointers).

All the identification and execution of built-in commands must be implemented in code in your builtin.c file. You should have at most two functions that processline() calls directly to implement your built-in commands. If you have two functions, one should answer the question "is the command described by this argv a built-in function" and the other one should do the built-in command. A single function would combine the functionality of the two into a single function.

Add the prototype for your built-in commands function(s) to your "builtin.h".

Note: if you are calling strcmp() in processline, you are not implementing this as requested and will lose points. Also, if you put definitions in ush.c to help you with built-in commands, you are also not doing as requested. All code implementing built-in commands must be in builtin.c. This includes global variables for built-in commands. These global variables must be static. No prototypes of "helper functions" in builtin.c should be in builtin.h, just the routines that are called from processline().

2) Implement exactly the following four built-in commands. (The brackets ([...]) in the descriptions do not appear in the real command, they just show that that part is optional.) Remember to check for errors and if an error is found, print an error message and then stop processing and return to processline().

- exit [value] - exit the shell with the value. If value is not given, exit with value 0.
  This built-in command should call the exit(3) library call. Hint: man 3 atoi
- envset NAME value
  that sets the environment variable of the same name to the given value.
- envunset NAME
  that removes the variable NAME from the current environment.
- cd [directory]
  Use chdir(2) to change the working directory of the shell. If the directory is not given, use the environment variable HOME. Any errors should report the error and then do nothing.