Background

To complete this assignment, you need a working shell from Homework 6. If your shell is not working or is not yet complete, please see me for assistance at your earliest convenience.

In the process of working on this assignment, you may find the following C functions useful. See the course web site for sample programs which illustrate their use.

- `strcpy()`
  String copy — copies a source string to a specified destination string.
- `strtoimax()`
  Converts a string value to an integer type.

Exercise 1. Add a *history* mechanism to your shell, borrowing some ideas from the Bash shell. When complete, your shell should meet the following specifications.

(a) Commands entered to your shell are numbered consecutively, starting with 1.

(b) Your shell should save a fixed number of the most recently-entered commands. This capacity should be controlled by the constant `MAX_COMMANDS`. During testing and for submission, this constant should have the value 10, although other values should be tried. Note: when you change this constant, no other code should have to be changed.

(c) Unlike Bash, your shell will lose all of its history upon termination. The history is only valid for as long as your shell is running. (Bash uses a file to store the history; in our shell, we will use primary memory.)

(d) In response to the "internal" command `history`, your shell should display a list of the most recent commands, up to a maximum of `MAX_COMMANDS`. Each line of this display should consist of the command number followed by the corresponding command. (An internal command is handled directly by the shell, as opposed to passing the work off to a child process.)

(e) To access a command from the history, the command line takes the form `!`n, where n is the number of the command, as it appears in the command history. For example, after working with your shell for awhile, it might happen that the 83rd command is `ls -la`. Rather than type this command, you can simply enter `!83`.

(f) Internally, your shell should save the command history in a 2-dimensional array:

```
char commandHistory[MAX_COMMANDS][MAX_LINE_LENGTH + 1];
```

Each row of this table will store one command. Viewing the rows arranged in a circle, a data structure similar to a queue can be constructed. Unlike a traditional queue, your command history will never overflow — as we continue to add commands, older commands will simply be overwritten.

(g) As your shell grows in size, introduce appropriate functions in order to maintain a reasonable level of modularity — we don't want to end up with one large `main()` function!

What to Submit

Place a copy of your completed shell program into a folder, then drag this folder onto the EIU submit icon. Be sure your name appears as a comment somewhere near the top of the file. Continue to provide internal documentation for your code.