Your shell should:

- Display a **prompt** when waiting for a new command.
- Have a working history.
- Search and launch the right executable (based on the **PATH** variable or using a relative or an absolute path).
- Use at most **one global variable** to indicate a received signal. Consider the implications: this approach ensures that your signal handler will not access your main data structure.
- Not interpret unclosed quotes or special characters which are not required by the subject such as \ (backslash) or ; (semicolon).
- Handle ‘ (single quote) which should prevent the shell from interpreting the meta-characters in the quoted sequence.
- Handle “ (double quote) which should prevent the shell from interpreting the meta-characters in the quoted sequence except for $ (dollar sign).
- Implement the following redirections:
  - < should redirect input.
  - > should redirect output.
  - << should be given a delimiter, then read the input until a line containing the delimiter is seen. However, it doesn’t have to update the history!
  - >> should redirect output in append mode.
- Implement **pipes** (| character). The output of each command in the pipeline is connected to the input of the next command via a pipe.
- Handle **environment variables** ($ followed by a sequence of characters) which should expand to their values.
- Handle $? which should expand to the exit status of the most recently executed foreground pipeline.
- Handle ctrl-C, ctrl-D, ctrl-\ which should behave like in bash.
- In interactive mode:
  - ctrl-C displays a new prompt on a new line.
  - ctrl-D exits the shell.
  - ctrl-\ does nothing.
- Your shell must implement the following **built-in** commands:
  - echo with option -n
  - cd with only a relative or absolute path
  - pwd with no options
  - export with no options
  - unset with no options
  - env with no options or arguments
  - exit with no options