Chapter III

Mandatory part

Program name	`minishell`
Turn in files	`Makefile, *.h, *.c`
Makefile	`NAME, all, clean, fclean, re`
Arguments	
External functs.	`readline, rl_clear_history, rl_on_new_line, rl_replace_line, rl_redisplay, add_history, printf, malloc, free, write, access, open, read, close, fork, wait, waitpid, wait3, wait4, signal, sigaction, sigemptyset, sigaddset, kill, exit, getcwd, chdir, stat, lstat, fstat, unlink, execve, dup, dup2, pipe, opendir, readdir, closedir, strerror, perror, isatty, ttyname, ttyslot, ioctl, getenv, tcsetattr, tcgetattr, tgetent, tgetflag, tgetnum, tgetstr, tgoto, tputs`
Libft authorized	`Yes`
Description	`Write a shell`

Your shell should:

- Display a **prompt** when waiting for a new command.
- Have a working **history**.

↗ Exec

- Search and launch the right executable (based on the `PATH` variable or using a relative or an absolute path).
- Avoid using more than **one global variable** to indicate a received signal. Consider the implications: this approach ensures that your signal handler will not access your main data structures.

```
Be careful. This global variable cannot provide any other
information or data access than the number of a received signal.
Therefore, using "norm" type structures in the global scope is
forbidden.
```

- 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 **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.

Redir

- 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.

Exit Statu

- Handle `ctrl-C`, `ctrl-D` and `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 **builtins**:
    - `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

Builtins

Chapter IV

Bonus part

Your program has to implement:

- `&&` and `||` with parenthesis for priorities.
- Wildcards `*` should work for the current working directory.

⚠ `The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.`