Minishell Project Complete Roadmap & Checklist

Project Constraints & Rules

Code Standards

- [ ] **Global Variables**: Only ONE global variable allowed (for signal tracking only)
- [ ] **Function Length**: Maximum 25 lines per function
- [ ] **Forbidden Constructs**:
  - [ ] No `for` loops
  - [ ] No `switch` statements
  - [ ] No ternary operators (`? :`)
- [ ] **Forbidden Functions**: No `snprintf`, `mkstemp`, etc.

Allowed Functions

- [ ] **Input/Output**: `readline`, `rl_clear_history`, `rl_on_new_line`, `rl_replace_line`, `rl_redisplay`, `add_history`, `printf`, `write`
- [ ] **Memory**: `malloc`, `free`
- [ ] **File Operations**: `access`, `open`, `read`, `close`, `unlink`
- [ ] **Process Management**: `fork`, `wait`, `waitpid`, `wait3`, `wait4`, `exit`, `execve`
- [ ] **Signals**: `signal`, `sigaction`, `sigemptyset`, `sigaddset`, `kill`
- [ ] **Directory**: `getcwd`, `chdir`, `stat`, `lstat`, `fstat`, `opendir`, `readdir`, `closedir`
- [ ] **File Descriptors**: `dup`, `dup2`, `pipe`
- [ ] **Terminal**: `isatty`, `ttyname`, `ttyslot`, `ioctl`, `tcsetattr`, `tcgetattr`
- [ ] **Terminal Capabilities**: `tgetent`, `tgetflag`, `tgetnum`, `tgetstr`, `tgoto`, `tputs`
- [ ] **Environment**: `getenv`
- [ ] **Error Handling**: `strerror`, `perror`
- [ ] **libft functions** (your own library)

Core Features Checklist

1. Basic Shell Functionality

- [ ] **Prompt Display**: Show prompt when waiting for new command
- [ ] **Command History**: Working history using arrow keys (Up/Down)
- [ ] **Command Execution**:
  - [ ] Execute with absolute path (`/bin/ls`)
  - [ ] Execute with relative path (`./program`)
  - [ ] Execute using PATH variable (`ls`, `cat`, etc.)

2. Quote Handling

- [ ] **Single Quotes (`'`)**:
  - [ ] Prevent interpretation of ALL metacharacters inside
  - [ ] `echo '$USER'` prints `$USER` literally
  - [ ] Handle empty single quotes `''`
  - [ ] Handle nested quotes properly
- [ ] **Double Quotes (")**:
  - [ ] Prevent interpretation of metacharacters EXCEPT `$`
  - [ ] Allow variable expansion inside double quotes
  - [ ] `echo "$USER"` prints the actual username
  - [ ] Handle empty double quotes `""`
- [ ] **Unclosed Quotes**: Should NOT be interpreted (error handling)
- [ ] **Mixed Quotes**: `echo hello'world'` should work correctly

3. Redirections

- [ ] **Input Redirection (<)**:
  - [ ] Basic: `grep hi < infile`
  - [ ] Multiple inputs: `grep hi < file1 < file2` (last one wins)
  - [ ] With missing files: proper error handling
  - [ ] Combined with pipes: `cat < infile | grep hello`
- [ ] **Output Redirection (>)**:
  - [ ] Basic: `ls > outfile`
  - [ ] Multiple outputs: `ls > file1 > file2` (last one wins)
  - [ ] Permission errors: handle gracefully
  - [ ] Combined with pipes: `echo hi > file | echo bye`
- [ ] **Append Redirection (>>)**:
  - [ ] Basic: `ls >> outfile`
  - [ ] Mixed with >: `ls > file1 >> file2`
  - [ ] Permission errors handling
- [ ] **Heredoc (<<)**:
  - [ ] Basic functionality: read until delimiter
  - [ ] Does NOT update history
  - [ ] Variable expansion in heredoc (without quotes)
  - [ ] No expansion with quoted delimiter: `cat << 'EOF'`
  - [ ] Multiple heredocs: `cat << EOF1 << EOF2`
  - [ ] Combined with other redirections

4. Pipes (`|`)

- [ ] **Basic Pipes**: `cat file | grep text | wc -l`
- [ ] **Error Handling**: `ls nonexistent | grep text`
- [ ] **Mixed with Redirections**: `cat < input | grep text > output`
- [ ] **Multiple Pipes**: Chain multiple commands correctly
- [ ] **Empty Pipe Segments**: Handle syntax errors

5. Environment Variables

- [ ] **Variable Expansion**:
  - [ ] Basic: `echo $USER`
  - [ ] In double quotes: `echo "$USER"`
  - [ ] NOT in single quotes: `echo '$USER'`
  - [ ] Multiple variables: `echo $HOME$USER`
  - [ ] Non-existent variables: expand to empty string
- [ ] **Special Variables**:
  - [ ] `$?`: Exit status of last command
  - [ ] Handle `$?` in expressions: `expr $? + $?`
  - [ ] Empty `$`: should print `$`
  - [ ] `$` at end of word: `echo hello$`

6. Signal Handling

- [ ] **Interactive Mode - Empty Prompt**:
  - [ ] `Ctrl-C`: Display new prompt on new line
  - [ ] `Ctrl-D`: Exit shell cleanly
  - [ ] `Ctrl-\`: Do nothing
- [ ] **Interactive Mode - With Input**:
  - [ ] `Ctrl-C`: Clear line, new prompt
  - [ ] `Ctrl-D`: Do nothing
  - [ ] `Ctrl-\`: Do nothing
- [ ] **During Command Execution**:
  - [ ] `Ctrl-C`: Interrupt command, return to prompt
  - [ ] `Ctrl-\`: Quit command (SIGQUIT)
  - [ ] `Ctrl-D`: Send EOF to command

7. Built-in Commands

echo

- [ ] Basic: `echo hello world`
- [ ] With `-n` flag: `echo -n hello` (no newline)
- [ ] Multiple `-n`: `echo -nnnnn hello` (treat as single -n)
- [ ] Mixed flags: `echo -nnnnnknn` should print `-nnnnnknn`
- [ ] Empty arguments: `echo "" " " hello`

cd

- [ ] Absolute path: `cd /home/user`
- [ ] Relative path: `cd ../..`
- [ ] Home directory: `cd` or `cd ~`
- [ ] Previous directory: `cd -`
- [ ] Update `PWD` and `OLDPWD`
- [ ] Error handling for non-existent directories

- [ ] Too many arguments: `cd dir1 dir2` (error)

pwd

- [ ] Print current working directory
- [ ] No options accepted
- [ ] Work even if `PWD` is unset

export

- [ ] Show all exports: `export` alone
- [ ] Set variable: `export VAR=value`
- [ ] Multiple exports: `export A=1 B=2`
- [ ] Export without value: `export VAR`
- [ ] Invalid names: `export 123=value` (error)
- [ ] Special characters: `export VAR-=value` (error)

unset

- [ ] Remove single variable: `unset VAR`
- [ ] Remove multiple: `unset VAR1 VAR2`
- [ ] Handle non-existent variables silently
- [ ] Critical variables: `unset PATH HOME PWD`

env

- [ ] Display all environment variables
- [ ] No arguments or options accepted

exit

- [ ] Exit with status: `exit 42`
- [ ] Without arguments: `exit` (status 0)
- [ ] Too many arguments: `exit 1 2` (error)
- [ ] Non-numeric argument: `exit hello` (error)
- [ ] Large numbers: handle overflow

Edge Cases & Advanced Tests

Syntax Error Handling

- [ ] Empty pipe: `|` or `| echo`
- [ ] Multiple redirections: `>>>` or `<<<`
- [ ] Invalid combinations: `>|`, `<|`, `|>`, `|<`
- [ ] Trailing operators: `echo hi |`, `echo hi >`
- [ ] Leading operators: `> file`, `| echo`

Complex Redirection Tests

- [ ] `cat < in1 < in2 > out1 > out2` (last ones win)

- [ ] `echo hi > file | echo bye` (redirection affects only first command)
- [ ] `< input1 command < input2` (position doesn't matter)
- [ ] Mixed heredoc and files: `cat << EOF < file`

Environment Variable Edge Cases

- [ ] `$USER$HOME$PATH` (concatenation)
- [ ] `echo $?HELLO` (exit code followed by text)
- [ ] `$EMPTY` (undefined variable)
- [ ] Command stored in variable: `export A='ls -la'` then `$A`

Special Character Handling

- [ ] Tabs as spaces: `\t/bin/ls\t-la`
- [ ] Multiple spaces between arguments
- [ ] Empty arguments: `echo "" " " test`
- [ ] Special chars in quotes: `echo "> >> < | & $ \\"`

Process & Exit Code Management

- [ ] `sleep 100 | ls` + Ctrl-C → `echo $?` should be 0
- [ ] Failed command: `/bin/ls nonexistent` → check exit code
- [ ] Multiple commands: track last exit code correctly
- [ ] Builtin vs external command exit codes

Memory Management

- [ ] No leaks in your code (readline may leak)
- [ ] Handle failed execve without leaks
- [ ] Clean up on signals
- [ ] Free all allocated resources on exit

Permission & Access Tests

- [ ] Execute non-executable files
- [ ] Write to read-only files
- [ ] Access denied directories
- [ ] Missing commands in PATH

Multi-line Input

- [ ] Handle: `echo a\n echo b\n echo c`
- [ ] Each command executes separately

Heredoc Advanced

- [ ] Nested heredocs with pipes
- [ ] Variable expansion vs literal (quoted delimiter)
- [ ] EOF in heredoc input
- [ ] Interrupting heredoc with signals

What NOT to Implement

- [ ] Backslash (`\`) escaping
- [ ] Semicolon (`;`) command separator
- [ ] && and `||` operators
- [ ] Wildcards (`*`, `?`, `[...]`)
- [ ] Command substitution with backticks
- [ ] Background processes (&)
- [ ] Job control
- [ ] Aliases
- [ ] Special parameters except `$?`

Testing Checklist

Basic Functionality Tests

- [ ] Empty command (just pressing Enter)
- [ ] Only spaces/tabs
- [ ] Very long command lines
- [ ] Commands with many arguments

Stress Tests

- [ ] Long pipe chains: `cat | cat | cat | cat | cat`
- [ ] Many redirections in one command
- [ ] Deeply nested quotes
- [ ] Very long environment variable values
- [ ] Rapid signal sending

Error Recovery

- [ ] Invalid command followed by valid command
- [ ] Syntax error followed by valid command
- [ ] Signal during heredoc input
- [ ] Full disk during output redirection

Environment Destruction Tests

- [ ] `unset PATH` - commands should fail
- [ ] `unset HOME` - cd should still work
- [ ] Empty environment startup
- [ ] Restore PATH and test command resolution order

Implementation Order Suggestion

1. **Basic REPL Loop**
    - Prompt display
    - Read input with readline

    - Basic parsing (tokenization)
2. **Simple Command Execution**
    - Fork/exec pattern
    - PATH resolution
    - Wait for child process
3. **Built-in Commands**
    - Start with `exit` and `echo`
    - Add `cd`, `pwd`
    - Implement `export`, `unset`, `env`
4. **Quote Parsing**
    - Single quotes first
    - Double quotes with expansion
    - Mixed quote handling
5. **Redirections**
    - Input redirection `<`
    - Output redirection `>`
    - Append `>>`
    - Heredoc `<<`
6. **Pipes**
    - Two-command pipes
    - Multi-command pipes
    - Pipes with redirections
7. **Signal Handling**
    - Basic signal setup
    - Interactive mode signals
    - Signal during command execution
8. **Environment Variables**
    - Basic expansion
    - Special variables (`$?`)
    - Export/unset integration
9. **Error Handling & Edge Cases**
    - Syntax errors
    - Permission errors
    - Memory leak fixes
10. **Polish & Optimization**
    - Code refactoring to meet constraints
    - Comprehensive testing
    - Norm compliance

Common Pitfalls to Avoid

1. **Signal Handler Complexity**: Keep it minimal, only set a flag
2. **Zombie Processes**: Always wait for children
3. **File Descriptor Leaks**: Close unused FDs in child processes
4. **Quote State Machines**: Handle all quote combinations
5. **PATH Resolution**: Check all directories in order
6. **Exit Status**: Update after EVERY command
7. **Memory Management**: Free everything, including on error paths
8. **Heredoc Implementation**: Don't update history
9. **Pipe Buffer Deadlocks**: Handle full pipe buffers
10. **Environment Modification**: Some commands modify the current shell's environment

Final Validation

- [ ] All mandatory tests pass
- [ ] No memory leaks (except readline)
- [ ] No crashes on any input
- [ ] Signals handled correctly
- [ ] Norm compliant (25 lines, no forbidden constructs)
- [ ] Only one global variable
- [ ] All built-ins working correctly
- [ ] Complex pipe/redirection combinations work
- [ ] Error messages similar to bash
- [ ] Clean exit on all conditions