Minishell Test Cases (Mandatory Part Only)

Based on subject version 9.0 - No bonus features

Subject Requirements Summary:

- Display prompt when waiting for command
- Working history
- Search and launch executables (PATH, relative, absolute)
- Handle quotes (' and ")
- Redirections: <, >, <<, >>
- Pipes (|)
- Environment variables ($VAR, $?)
- Signals: ctrl-C, ctrl-D, ctrl-\
- Builtins: echo -n, cd, pwd, export, unset, env, exit

---

1. Basic Command Execution & PATH

Test 1: Command with absolute path

```bash
$ /bin/echo hello
hello
$ echo $?
0
```

Test 2: Command with relative path

```bash
$ ./minishell
# (should launch another minishell instance if compiled)
```

Test 3: Command from PATH

```bash
```

```
$ ls
# Expected: List current directory contents
$ echo $?
0
```

Test 4: Command not found

```bash
$ invalidcommand
invalidcommand: command not found
$ echo $?
127
```

Test 5: Empty command

```bash
$
# Expected: Just show new prompt, no error
```

Test 6: Only spaces/tabs

```bash
$
# Expected: Just show new prompt, no error
```

---

2. Arguments

Test 7: Multiple arguments

```bash
$ echo hello world 42 school
hello world 42 school
```

Test 8: Arguments with special characters (not operators)

```bash
```

```
$ echo hello-world test_file
hello-world test_file
```

---

3. Single Quotes (No Interpretation)

Test 9: Single quotes preserve everything literally

```
bash

$ echo 'hello   world'
hello   world
```

Test 10: $ in single quotes (no expansion)

```
bash

$ echo '$USER'
$USER
```

Test 11: $ in single quotes with text

```
bash

$ echo 'hello $USER world'
hello $USER world
```

Test 12: Empty single quotes

```
bash

$ echo ''


$ echo hello '' world
hello  world
```

Test 13: Single quotes with special chars

```
bash
```

```
$ echo 'hello | world > test'
hello | world > test
```

---

4. Double Quotes (Expand $ only)

Test 14: Double quotes preserve spaces

```bash
$ echo "hello   world"
hello   world
```

Test 15: $ expands in double quotes

```bash
$ echo "hello $USER"
hello <your_username>
```

Test 16: $? expands in double quotes

```bash
$ ls
$ echo "exit code: $?"
exit code: 0
```

Test 17: Empty double quotes

```bash
$ echo ""

$ echo hello "" world
hello  world
```

Test 18: Pipes/redirects in double quotes (literal)

```bash
```

```
$ echo "hello | world"
hello | world
```

Test 19: Mix single and double quotes

```bash
$ echo "hello '$USER' world"
hello '$USER' world

$ echo 'hello "$USER" world'
hello "$USER" world
```

Test 20: Adjacent quotes

```bash
$ echo "hello"world
helloworld

$ echo 'hello'"world"
helloworld
```

---

5. Environment Variables

Test 21: Simple variable expansion

```bash
$ echo $USER
<your_username>
```

Test 22: Variable with text

```bash
```

```
$ echo $USER_test
# Expected: empty (variable doesn't exist)

$ echo ${USER}_test
<your_username>_test
```

Test 23: Multiple variables

```
bash

$ echo $USER $HOME $PATH
<username> <home_path> <path_values>
```

Test 24: Undefined variable

```
bash

$ echo $NONEXISTENT
# Expected: empty line (no output)
```

Test 25: $ alone

```
bash

$ echo $
$
```

Test 26: $? - exit status

```
bash

$ ls
$ echo $?
0

$ /bin/false
$ echo $?
1

$ invalidcmd
$ echo $?
127
```

Test 27: Variable in quotes

```bash
$ echo '$USER'
$USER

$ echo "$USER"
<your_username>
```

---

6. Redirections - Input (<)

Test 28: Basic input redirection

```bash
$ echo "test content" > infile
$ cat < infile
test content
```

Test 29: Input file doesn't exist

```bash
$ cat < nonexistent
bash: nonexistent: No such file or directory
$ echo $?
1
```

Test 30: Input redirection position

```bash
$ < infile cat
test content

$ cat < infile
test content
# Both should work the same
```

---

7. Redirections - Output (>)

Test 31: Basic output redirection

bash

```
$ echo hello > outfile
$ cat outfile
hello
```

Test 32: Output overwrites file

bash

```
$ echo first > outfile
$ echo second > outfile
$ cat outfile
second
```

Test 33: Output to /dev/null

bash

```
$ echo hello > /dev/null
# No output visible
$ echo $?
0
```

Test 34: Permission denied

bash

```
$ echo test > /root/file
bash: /root/file: Permission denied
$ echo $?
1
```

Test 35: Multiple output redirections (last wins)

bash

```
$ echo test > file1 > file2
$ cat file2
test
$ cat file1
# file1 should be empty or created but empty
```

---

8. Redirections - Append (>>)

Test 36: Basic append

bash

```
$ echo first > file
$ echo second >> file
$ cat file
first
second
```

Test 37: Append to new file

bash

```
$ echo hello >> newfile
$ cat newfile
hello
```

Test 38: Multiple appends

bash

```
$ echo line1 > file
$ echo line2 >> file
$ echo line3 >> file
$ cat file
line1
line2
line3
```

---

9. Redirections - Heredoc (<<)

Test 39: Basic heredoc

```bash
$ cat << EOF
> line1
> line2
> EOF
line1
line2
```

Test 40: Heredoc with delimiter

```bash
$ cat << END
> hello
> world
> END
hello
world
```

Test 41: Heredoc with variable expansion

```bash
$ cat << EOF
> Hello $USER
> EOF
Hello <your_username>
```

Test 42: Heredoc should NOT update history

```bash
# The heredoc content lines should NOT appear in history
# Only the initial command should be in history
```

Test 43: Multiple heredocs

```bash
```

```
$ cat << EOF << END
> test
> EOF
# Behavior depends on implementation
# Usually last heredoc is used
```

---

10. Pipes (|)

Test 44: Simple pipe

bash

```
$ echo hello | cat
hello
```

Test 45: Pipe with grep

bash

```
$ echo -e "hello\nworld\ntest" | grep world
world
```

Test 46: Multiple pipes

bash

```
$ echo "test" | cat | cat | cat
test
```

Test 47: Pipe with wc

bash

```
$ echo "one two three" | wc -w
3
```

Test 48: ls piped to grep

bash

```
$ ls | grep minishell
minishell
# (assuming minishell file exists)
```

Test 49: Pipe with exit status

```bash
$ ls | grep nonexistent
$ echo $?
1
# Exit status should be from last command in pipeline
```

Test 50: Failed command in pipe

```bash
$ invalidcmd | cat
invalidcmd: command not found
$ echo $?
127
# But should still show error for first command
```

---

11. Combined Redirections and Pipes

Test 51: Pipe with output redirection

```bash
$ echo hello | cat > outfile
$ cat outfile
hello
```

Test 52: Pipe with input redirection

```bash
$ echo test > infile
$ < infile cat | grep test
test
```

Test 53: Multiple redirections with pipe

```bash
$ < infile cat | grep test > outfile
$ cat outfile
test
```

Test 54: Heredoc with pipe

```bash
$ cat << EOF | grep hello
> hello world
> test
> EOF
hello world
```

---

12. Built-in: echo

Test 55: echo without arguments

```bash
$ echo
```

Test 56: echo with text

```bash
$ echo hello world
hello world
```

Test 57: echo -n (no newline)

```bash
$ echo -n hello
hello$ # (no newline, prompt on same line)
```

Test 58: echo -n -n -n (multiple -n)

```bash
$ echo -n -n -n hello
hello$ # (still no newline)
```

Test 59: echo -nnn (combined n's)

```bash
$ echo -nnnnn hello
hello$ # (no newline)
```

Test 60: echo with invalid flag

```bash
$ echo -x hello
-x hello
# -x is treated as argument, not flag
```

Test 61: echo -n with variables

```bash
$ echo -n $USER
<username>$ # (no newline)
```

---

13. Built-in: cd

Test 62: cd to home (no args)

```bash
$ cd
$ pwd
/home/<username>
# Or whatever $HOME is
```

Test 63: cd to absolute path

```bash
```

```
$ cd /tmp
$ pwd
/tmp
```

Test 64: cd to relative path

```bash
$ cd ..
$ pwd
# Expected: parent directory
```

Test 65: cd to non-existent directory

```bash
$ cd /nonexistent
cd: /nonexistent: No such file or directory
$ echo $?
1
```

Test 66: cd with too many arguments

```bash
$ cd /tmp /home
cd: too many arguments
$ echo $?
1
```

Test 67: cd to directory without permission

```bash
$ cd /root
cd: /root: Permission denied
$ echo $?
1
```

Test 68: cd with variable

```bash
```

```
$ cd $HOME
$ pwd
/home/<username>
```

Test 69: cd to -

```
bash

$ cd /tmp
$ cd -
# Expected: might not be implemented (subject says "only relative or absolute path")
# If not implemented: cd: -: invalid option or similar error
```

---

14. Built-in: pwd

Test 70: pwd basic

```
bash

$ pwd
/current/working/directory
```

Test 71: pwd after cd

```
bash

$ cd /tmp
$ pwd
/tmp
```

Test 72: pwd with arguments (should ignore)

```
bash

$ pwd hello world
/current/working/directory
# Arguments should be ignored, no options supported
```

Test 73: pwd with redirections

```bash
$ pwd > file
$ cat file
/current/working/directory
```

---

15. Built-in: export

Test 74: export new variable

```bash
$ export TEST=hello
$ echo $TEST
hello
```

Test 75: export without value

```bash
$ export TEST
# Variable marked for export (if it exists)
```

Test 76: export multiple variables

```bash
$ export A=1 B=2 C=3
$ echo $A $B $C
1 2 3
```

Test 77: export with quotes

```bash
$ export TEST="hello world"
$ echo $TEST
hello world
```

Test 78: export without arguments (list all)

```bash
$ export
# Expected: list of all exported variables
# Format similar to bash: declare -x VAR="value"
```

Test 79: export invalid identifier

```bash
$ export 123=test
bash: export: `123=test': not a valid identifier
$ echo $?
1
```

Test 80: export with special characters

```bash
$ export TEST-VAR=hello
bash: export: `TEST-VAR=hello': not a valid identifier
```

Test 81: export variable with $

```bash
$ export NEW=$USER
$ echo $NEW
<username>
```

---

16. Built-in: unset

Test 82: unset existing variable

```bash
```

```
$ export TEST=hello
$ echo $TEST
hello
$ unset TEST
$ echo $TEST
# Expected: empty
```

Test 83: unset non-existent variable

```
bash

$ unset NONEXISTENT
# No error, just returns
$ echo $?
0
```

Test 84: unset multiple variables

```
bash

$ export A=1 B=2 C=3
$ unset A B C
$ echo $A $B $C
# Expected: empty empty empty
```

Test 85: unset PATH

```
bash

$ unset PATH
$ ls
bash: ls: No such file or directory
# (commands won't be found without PATH)
```

Test 86: unset invalid identifier

```
bash

$ unset 123
bash: unset: `123': not a valid identifier
$ echo $?
1
```

17. Built-in: env

Test 87: env with no arguments

bash

```
$ env
# Exp
```