Comprehensive Minishell Lexer & Builtin Test Cases

LEXER TEST CASES

1. Basic Tokenization

Simple words

```bash
echo
ls
cat
pwd
hello
test123
```

Words with special characters

```bash
file.txt
file_name
file-name
/path/to/file
./relative/path
../parent/path
~/.bashrc
file123.txt
test_file-2.sh
```

2. Operators

Single character operators

```bash
|
>
<
&
;
```

Multi-character operators

```bash
>>
<<
&&
||
```

Operators with spacing variations

```bash
echo hello|cat
echo hello | cat
echo hello |cat
echo hello| cat
echo>file.txt
echo >file.txt
echo> file.txt
echo > file.txt
cat<input.txt
cat <input.txt
cat< input.txt
cat < input.txt
```

3. Quote Handling

Single quotes - no expansion

```bash
echo 'hello world'
echo '$USER'
echo '$HOME/test'
echo 'single"double'
echo """
echo 'a"b"c'
echo 'test | pipe'
echo 'test > redirect'
```

Double quotes - with expansion

```bash
```

```
echo "hello world"
echo "$USER"
echo "$HOME/test"
echo "double'single"
echo """""""
echo "a""b""c"
echo "test $USER test"
echo "$USER$HOME"
echo "test | pipe"
echo "test > redirect"
```

Mixed quotes

bash

```
echo "hello"'world'
echo 'single'"double"
echo "test"'$USER'"more"
echo '$USER'"$HOME"
echo "test'nested'test"
echo 'test"nested"test'
```

Unclosed quotes (should error or wait for closing)

bash

```
echo "hello
echo 'world
echo "test'mixed
echo 'test"mixed
```

Empty quotes

bash

```
echo ""
echo "
echo """"
echo ""
echo """"
echo """
```

Quotes in middle of words

```bash
echo hel"lo"world
echo test'123'end
echo start"middle"end'final'
echo $USER"test"$HOME
```

4. Variable Expansion

Basic expansion

```bash
echo $USER
echo $HOME
echo $PATH
echo $PWD
echo $OLDPWD
echo $?
echo $_
```

Expansion with text

```bash
echo prefix$USERsuffix
echo $USER$HOME
echo test$USERtest
echo $USER/$HOME/test
```

Non-existent variables

```bash
echo $NONEXISTENT
echo $DOES_NOT_EXIST
echo $
echo test$test
```

Special cases

```bash
echo $$
echo $?
echo $0
echo $1
echo $@
echo $*
echo $#
```

Variables in quotes

```bash
echo "$USER"
echo '$USER'
echo "$USER$HOME"
echo '$USER$HOME'
echo "test $USER test"
echo 'test $USER test'
```

Edge cases

```bash
echo $
echo $ test
echo test$
echo $123
echo $123test
echo $test123
echo ${USER}
echo ${}
```

5. Whitespace Handling

Multiple spaces

```bash
echo    hello    world
ls   -l   -a
cat    file.txt
```

Tabs

```bash
echo	hello	world
ls	-l	-a
```

Mixed whitespace

```bash
echo    hello   world
ls  -l  -a
```

Leading/trailing whitespace

```bash
  echo hello
echo hello
  ls -la
 echo test
```

No whitespace

```bash
echo"hello"
cat<file.txt>output.txt
ls|grep test|wc -l
```

6. Special Characters

Backslash (if supported)

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

Wildcards (if supported)

bash

```
echo *
echo *.txt
echo test*
echo *test
echo test*.txt
ls *.c
```

Parentheses (if subshells supported)

bash

```
(echo hello)
( echo hello )
((echo nested))
(echo one; echo two)
```

7. Comments (if supported)

bash

```
echo hello # this is a comment
# this is a full line comment
echo test # comment | not a pipe
# echo this should not run
```

8. Complex Tokenization

Multiple operators

bash

```
echo hello | cat | cat | wc -l
cat < input.txt | grep test > output.txt
echo test && echo success || echo fail
```

Quotes with operators

bash

```
echo "hello | world"
echo 'test > file'
echo "test && success"
cat "file > name.txt"
```

Variables with operators

```bash
echo $USER | cat
cat $HOME/.bashrc | grep alias
echo $? && echo success
```

9. Edge Cases in Lexing

Empty input

```
```

Only spaces/tabs

```
```

Only operators

```bash
|
||
&&
><
```

Malformed operators

```bash
|||
&&&
>>>
<<<
```

Consecutive quotes

```bash
echo """"test""""
echo ""test""
echo """""""
```

Quote combinations

```bash
echo "test'test"test
echo 'test"test'test
echo """""
```

BUILTIN TEST CASES

1. echo

Basic echo

```bash
echo
echo hello
echo hello world
echo "hello world"
echo 'hello world'
```

-n flag (no newline)

```bash
echo -n hello
echo -n "hello world"
echo -n hello world
echo -n -n hello
echo -nnnn hello
echo -n -n -n hello world
```

Invalid flags (should print as text)

```bash
echo -a hello
echo -x test
echo -N hello
echo - n hello
```

With variables

```bash
echo $USER
echo $HOME
echo $PATH
echo $?
echo "$USER is logged in"
echo '$USER is logged in'
```

Special characters

```bash
echo \$USER
echo \\
echo \"test\"
echo \n\t\r
```

Multiple arguments

```bash
echo one two three four five
echo "arg1" "arg2" "arg3"
echo arg1 'arg2' "arg3"
```

Edge cases

```bash
```

```
echo ""
echo "
echo -n
echo -n ""
echo -n -n
echo $NONEXISTENT
```

2. cd

Basic cd

bash

```
cd /tmp
cd ~
cd
cd /
cd /usr/local/bin
```

Relative paths

bash

```
cd ..
cd ../..
cd ./test
cd test/subdir
```

Special directories

bash

```
cd ~
cd ~/Documents
cd -
cd $HOME
cd "$HOME/test"
```

Error cases

bash

```
cd /nonexistent
cd /root
cd file.txt
cd ""
cd "  "
```

Multiple arguments (should error)

bash

```
cd /tmp /var
cd ~ /tmp
cd one two three
```

With quotes

bash

```
cd "test dir"
cd 'test dir'
cd "/tmp"
cd '/tmp'
```

Complex paths

bash

```
cd ../.././test/../tmp
cd /tmp/./test
cd ~/./Documents
```

Edge cases

bash

```
cd .
cd ..
cd /
cd //
cd ///
```

PWD updates

```bash
cd /tmp
echo $PWD
echo $OLDPWD
cd -
echo $PWD
echo $OLDPWD
```

3. pwd

Basic pwd

```bash
pwd
cd /tmp && pwd
cd ~ && pwd
```

With options (might not be supported)

```bash
pwd -L
pwd -P
pwd --help
```

After directory changes

```bash
pwd
cd /tmp
pwd
cd ..
pwd
cd ~
pwd
```

After directory deletion

```bash
```

```
# Setup in another terminal: mkdir -p /tmp/test && cd /tmp/test
pwd
# Delete /tmp/test in another terminal
pwd
```

Edge cases

```bash
pwd extra arguments
pwd | cat
pwd > output.txt
```

4. export

Basic export

```bash
export VAR=value
export VAR="value"
export VAR='value'
export PATH=/bin:/usr/bin
```

Multiple exports

```bash
export VAR1=value1 VAR2=value2
export VAR1=val1 VAR2=val2 VAR3=val3
```

Export without value (declare as exported)

```bash
export VAR
export PATH
export USER
```

No arguments (print all exported vars)

```bash
```

```
export
```

Complex values

bash

```
export VAR="value with spaces"
export PATH=$PATH:/new/path
export TEST="$USER:$HOME"
export VAR='$USER'
export MULTI="line1
line2"
```

Invalid variable names

bash

```
export 123VAR=value
export VAR-NAME=value
export VAR NAME=value
export =value
export "INVALID VAR"=value
```

Empty values

bash

```
export VAR=
export VAR=""
export VAR=''
```

Overwriting variables

bash

```
export VAR=old
export VAR=new
echo $VAR
```

Special characters in values

bash

```
export VAR="test|pipe"
export VAR="test>redirect"
export VAR="test;semicolon"
export VAR="test\$dollar"
```

Edge cases

```bash
export VAR=value1 VAR=value2
export ""
export ''
export VAR
export NONEXISTENT
```

5. unset

Basic unset

```bash
export VAR=value
unset VAR
echo $VAR
```

Multiple unsets

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

Unset special variables

```bash
unset PATH
unset HOME
unset USER
unset PWD
```

Non-existent variables

```bash
unset NONEXISTENT
unset FAKE_VAR
```

No arguments (should do nothing or error)

```bash
unset
```

Invalid variable names

```bash
unset 123VAR
unset VAR-NAME
unset "INVALID VAR"
```

Edge cases

```bash
unset ""
unset ''
unset VAR1 VAR2 VAR3 NONEXISTENT VAR4
```

After unset

```bash
export TEST=value
echo $TEST
unset TEST
echo $TEST
export TEST=newvalue
echo $TEST
```

6. env

Basic env

```bash
```

```
env
```

With arguments (might not be supported)

bash

```
env | grep USER
env | sort
```

After modifications

bash

```
export TEST=123
env | grep TEST
unset TEST
env | grep TEST
```

Empty environment (edge case)

bash

```
# This is more for testing, not typical usage
env -i ./minishell
env
```

Redirection

bash

```
env > env_output.txt
env | cat
```

7. exit

Basic exit

bash

```
exit
exit 0
exit 1
exit 42
```

Edge cases

```bash
exit 255
exit 256
exit -1
exit 999999
```

Invalid arguments

```bash
exit hello
exit 123abc
exit abc123
exit "42"
exit '42'
```

Multiple arguments (should error)

```bash
exit 1 2 3
exit 0 extra
```

Exit with expressions

```bash
exit $?
exit $(echo 42)
```

Exit status propagation

```bash
```

```
false
exit
# Check exit status in parent shell
```

In subshells

```bash
(exit 42)
echo $?
```

COMBINED LEXER + BUILTIN TESTS

1. Builtins with Complex Quoting

```bash
echo "USER is $USER"
echo 'USER is $USER'
export VAR="test'quote"
export VAR='test"quote'
cd "test dir"
cd 'test dir'
```

2. Builtins with Operators

```bash
echo hello | cat
pwd | cat
env | grep USER
export TEST=123 && echo $TEST
cd /tmp && pwd
cd /fake || echo "failed"
```

3. Builtins with Redirections

```bash
```

```
echo hello > output.txt
pwd > pwd_output.txt
env > env_output.txt
export TEST=123 > /dev/null
cd /tmp > /dev/null
```

4. Multiple Builtins

bash

```
export TEST=hello; echo $TEST
cd /tmp; pwd; cd -; pwd
export A=1; export B=2; echo $A $B
unset A; unset B; echo $A $B
```

5. Builtins with Variables

bash

```
echo $HOME
cd $HOME
export PATH=$PATH:/new/path
cd "$HOME/Documents"
unset HOME
cd $HOME
```

6. Edge Case Combinations

bash

```
echo "" | cat
export VAR= | cat
cd ""
pwd | | cat
exit | cat
```

7. Whitespace Madness

bash

```
  echo   "hello"  |  cat
 export   VAR=test
cd      /tmp
   pwd
```

8. Quote Escaping in Builtins

bash

```
echo "test\"quote"
echo 'test\'quote'
export VAR="value\"with\"quotes"
cd "dir\"name"
```

9. Variable Expansion in Builtins

bash

```
echo $USER$HOME
export NEW=$OLD
cd $HOME/$USER
echo "$USER is at $HOME"
```

10. Stress Tests

bash

```
echo "test" | cat | cat | cat | grep test | cat | wc -l
export A=1 && export B=2 && export C=3 && echo $A $B $C
cd /tmp && pwd && cd ~ && pwd && cd - && pwd
export VAR="very long value with many words and spaces and special chars !@#$%^&*()"
```

SYSTEMATIC TEST SCRIPT

bash

```bash
#!/bin/bash

# Test script to compare bash vs minishell

TESTS=(
    "echo hello"
    "echo 'hello world'"
    'echo "hello world"'
    "echo \$USER"
    "pwd"
    "cd /tmp && pwd"
    "export TEST=value && echo \$TEST"
    "unset PATH && echo \$PATH"
    "exit 42"
)

for test in "${TESTS[@]}"; do
    echo "Testing: $test"

    # Run in bash
    bash -c "$test" > bash.out 2>&1
    bash_exit=$?

    # Run in minishell
    ./minishell -c "$test" > mini.out 2>&1
    mini_exit=$?

    # Compare outputs
    if diff bash.out mini.out > /dev/null && [ $bash_exit -eq $mini_exit ]; then
        echo "✓ PASS"
    else
        echo "✗ FAIL"
        echo "Bash output:"
        cat bash.out
        echo "Bash exit: $bash_exit"
        echo "Minishell output:"
        cat mini.out
        echo "Minishell exit: $mini_exit"
    fi
    echo "---"
done
```

```
rm bash.out mini.out
```

LEXER-SPECIFIC EDGE CASES

Token Boundary Detection

bash

```
echo"test"
echo'test'
cat<file.txt
cat>file.txt
ls|grep
echo&&pwd
cd||ls
```

Maximum Token Length

bash

```
echo verylongtokenwithoutspacesverylongtokenwithoutspacesverylongtokenwithoutspaces...
```

Unicode and Special Characters (if supported)

bash

```
echo "日本語"
echo "emoji: 🚀"
echo "special: àéîôù"
```

Null Bytes (should handle gracefully)

bash

```
# Test with files containing null bytes
```

Maximum Argument Count

bash

```
echo 1 2 3 4 5 ... 1000
```

This comprehensive test suite will thoroughly exercise your lexer and builtin implementations!