Both tests are also found in the zip folder as text files:

Command Line:

```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run list
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.03s
     Running `target\debug\PL-Final.exe list`
Available Commands:
help,
list,
tokenize,
parse,
execute
```

```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run help
   Compiling PL-Final v0.1.0 (C:\Users\megan\RustroverProjects\PL-Final)
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 2.91s
     Running `target\debug\PL-Final.exe help`
help                Display the help message for all commands.
help [command]      Display the help message for the specified command.
list                Display the list of supported commands.
tokenize            Display the lexical analysis on a given file.
parse               Display the parsed tree given an input file.
execute             Display the executed code of a given input file.
```

---

Test1:

```
func inc(n) [
    return n + 1;
]

func main() [
    let x;
    x = inc(4);
    print x;
    return x;
]
```

```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run tokenize Test1
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.03s
     Running `target\debug\PL-Final.exe tokenize Test1`
FUNC ID("inc") ( ID("n") ) [ RETURN ID("n") + INT(1) ; ] FUNC ID("main") ( ) [ LET ID("x") ; ID("x") = ID("inc") ( INT(4) ) ;
PRINT ID("x") ; RETURN ID("x") ; ] EOI
```

```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run parse Test1
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.02s
     Running `target\debug\PL-Final.exe parse Test1`
--- AST (MTree) ---
PROGRAM
  FUNCTION
    IDENTIFIER("inc")
    PARAM_LIST
      PARAMETER
        IDENTIFIER("n")
    BLOCK
      STATEMENT
        RETURN
          OPERATOR("+")
            IDENTIFIER("n")
            INT_LITERAL(1)
  FUNCTION
    IDENTIFIER("main")
    PARAM_LIST
    BLOCK
      STATEMENT
        LET
          IDENTIFIER("x")
      STATEMENT
        ASSIGN
          IDENTIFIER("x")
          FUNCTION_CALL("inc")
            IDENTIFIER("inc")
            INT_LITERAL(4)
      STATEMENT
        PRINT
          IDENTIFIER("x")
      STATEMENT
        RETURN
          IDENTIFIER("x")
```

```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run execute Test2
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.02s
     Running `target\debug\PL-Final.exe execute Test2`
--- ANALYZING ---
--- RUNNING PROGRAM ---
0
120
--- DONE ---
```

Test2:

```
func factorial_recursion(n) [
    if n < 2 [
        return 1;
    ]
    else [
        return n * factorial_recursion(n - 1);
    ]
]

func factorial_loop(n) [
    let p;
    p = n;

    while n > 0 [
        n = n - 1;
        p = p * n;
    ]
    return p;
]

func main() [
    let n;
    n = 5;
    print factorial_loop(n);
    print factorial_recursion(n);
]
```

```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run tokenize Test2
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.13s
     Running `target\debug\PL-Final.exe tokenize Test2`
FUNC ID("factorial_recursion") ( ID("n") ) [ IF ID("n") < INT(2) [ RETURN INT(1) ; ] ELSE [ RETURN ID("n")
* ID("factorial_recursion") ( ID("n") - INT(1) ) ; ] ] FUNC ID("factorial_loop") ( ID("n") ) [ LET ID("p")
; ID("p") = ID("n") ; WHILE ID("n") > INT(0) [ ID("n") = ID("n") - INT(1) ; ID("p") = ID("p") * ID("n") ; ]
 RETURN ID("p") ; ] FUNC ID("main") ( ) [ LET ID("n") ; ID("n") = INT(5) ; PRINT ID("factorial_loop") ( ID(
"n") ) ; PRINT ID("factorial_recursion") ( ID("n") ) ; ] EOI
```

```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run execute Test2
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.03s
     Running `target\debug\PL-Final.exe execute Test2`
--- ANALYZING ---
--- RUNNING PROGRAM ---
0
120
--- DONE ---
```

```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run parse Test2
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.01s
     Running `target\debug\PL-Final.exe parse Test2`
--- AST (MTree) ---
PROGRAM
  FUNCTION
    IDENTIFIER("factorial_recursion")
    PARAM_LIST
      PARAMETER
        IDENTIFIER("n")
    BLOCK
      STATEMENT
        IF
          OPERATOR("<")
            IDENTIFIER("n")
            INT_LITERAL(2)
          BLOCK
            STATEMENT
              RETURN
                INT_LITERAL(1)
          BLOCK
            STATEMENT
              RETURN
                OPERATOR("*")
                  IDENTIFIER("n")
                  FUNCTION_CALL("factorial_recursion")
                    IDENTIFIER("factorial_recursion")
                    OPERATOR("-")
                      IDENTIFIER("n")
                      INT_LITERAL(1)
```

```
  FUNCTION
    IDENTIFIER("factorial_loop")
    PARAM_LIST
      PARAMETER
        IDENTIFIER("n")
    BLOCK
      STATEMENT
        LET
          IDENTIFIER("p")
      STATEMENT
        ASSIGN
          IDENTIFIER("p")
          IDENTIFIER("n")
      STATEMENT
        WHILE
          OPERATOR(">")
            IDENTIFIER("n")
            INT_LITERAL(0)
          BLOCK
            STATEMENT
              ASSIGN
                IDENTIFIER("n")
                OPERATOR("-")
                  IDENTIFIER("n")
                  INT_LITERAL(1)
            STATEMENT
              ASSIGN
                IDENTIFIER("p")
                OPERATOR("*")
                  IDENTIFIER("p")
                  IDENTIFIER("n")
```

```
      STATEMENT
        RETURN
          IDENTIFIER("p")
  FUNCTION
    IDENTIFIER("main")
    PARAM_LIST
    BLOCK
      STATEMENT
        LET
          IDENTIFIER("n")
      STATEMENT
        ASSIGN
          IDENTIFIER("n")
          INT_LITERAL(5)
      STATEMENT
        PRINT
          FUNCTION_CALL("factorial_loop")
            IDENTIFIER("factorial_loop")
            IDENTIFIER("n")
      STATEMENT
        PRINT
          FUNCTION_CALL("factorial_recursion")
            IDENTIFIER("factorial_recursion")
            IDENTIFIER("n")
```