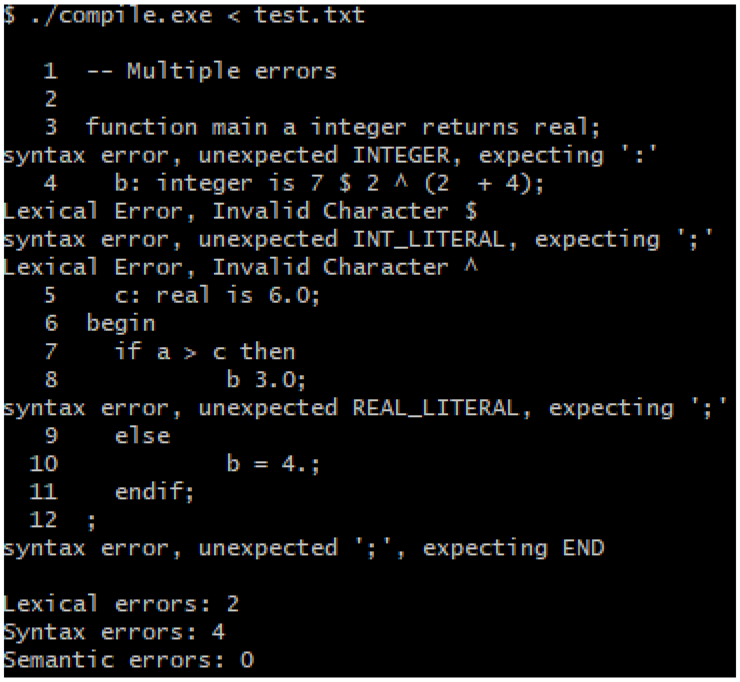
My approach to this project was first figuring out how parser.y worked. This was my first time working with the parser, so it was quite fun to mess around and see how it worked. All I really needed to know was within the code already, it was just a matter of understanding how all of the pieces fit together. I had quite a bit of trouble figure out the 0 or more occurrences, but once I found how it worked it was easy enough to implement it. I tried for quite a while to eliminate all of the shift/reduce conflicts but I couldn't solve them all. The -Wcounterexamples command did help a bit but sometimes it was too complicated to understand with my current level of knowledge.

Some improvements to the program would be better error detection. For instance, in test case #11 when the variable is missing its identifier, it said "expecting IDENTIFIER or BEGIN_" I'm wondering if it is possible for the program to take the rest of the line and recognize that it is a variable that is missing its identifier rather than the possibility of it being begin. Essentially, it is only missing the identifier but everything else is syntactically correct, don't say "or BEGIN_".

Test cases in the form of figures of the program running and parsing each text file starts below:

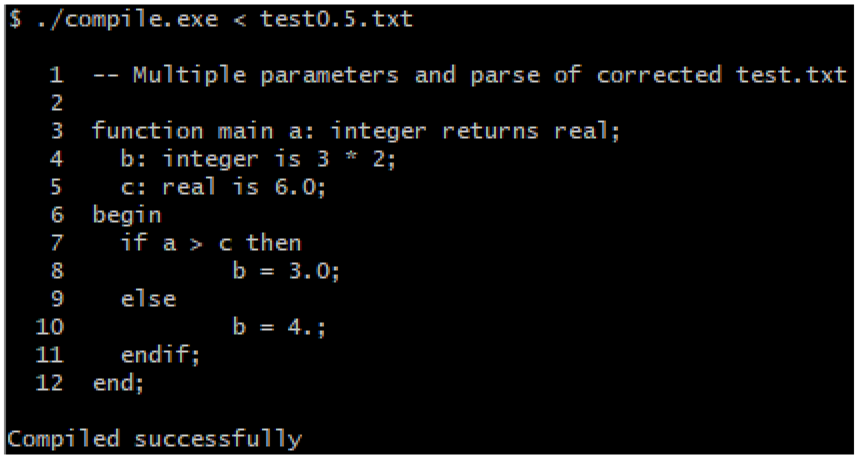


```
$ ./compile.exe < test.txt

   1  -- Multiple errors
   2
   3  function main a integer returns real;
syntax error, unexpected INTEGER, expecting ':'
   4    b: integer is 7 $ 2 ^ (2  + 4);
Lexical Error, Invalid Character $
syntax error, unexpected INT_LITERAL, expecting ';'
Lexical Error, Invalid Character ^
   5    c: real is 6.0;
   6  begin
   7    if a > c then
   8            b 3.0;
syntax error, unexpected REAL_LITERAL, expecting ';'
   9    else
  10            b = 4.;
  11    endif;
  12  ;
syntax error, unexpected ';', expecting END

Lexical errors: 2
Syntax errors: 4
Semantic errors: 0
```

Figure 1. Test case #1. Parameter missing colon, invalid characters, missing equal sign, and missing end.

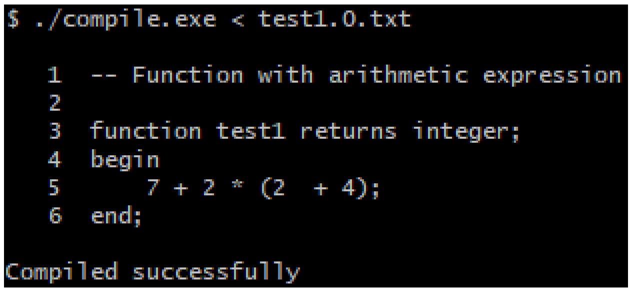


```
$ ./compile.exe < test0.5.txt

   1  -- Multiple parameters and parse of corrected test.txt
   2
   3  function main a: integer returns real;
   4    b: integer is 3 * 2;
   5    c: real is 6.0;
   6  begin
   7    if a > c then
   8            b = 3.0;
   9    else
  10            b = 4.;
  11    endif;
  12  end;

Compiled successfully
```

Figure 2. Test case #2. Corrected test case 1 which showcases if statements and integer/real variables

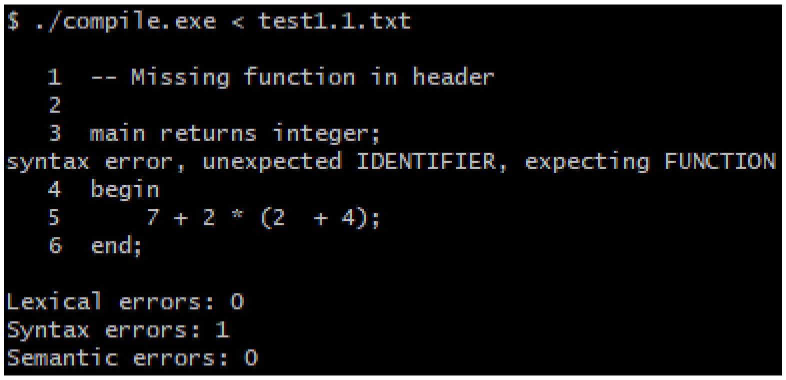


```
$ ./compile.exe < test1.0.txt

   1  -- Function with arithmetic expression
   2
   3  function test1 returns integer;
   4  begin
   5      7 + 2 * (2  + 4);
   6  end;

Compiled successfully
```

Figure 3. Test case #3. Valid function.

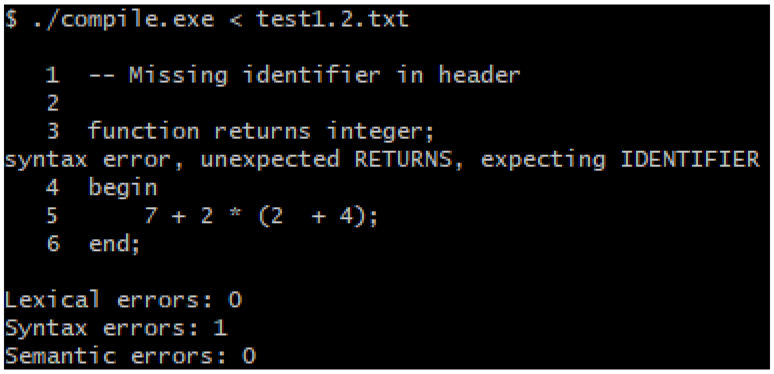


```
$ ./compile.exe < test1.1.txt

   1  -- Missing function in header
   2
   3  main returns integer;
syntax error, unexpected IDENTIFIER, expecting FUNCTION
   4  begin
   5      7 + 2 * (2  + 4);
   6  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 4. Test case #4. Missing function keyword in function header.

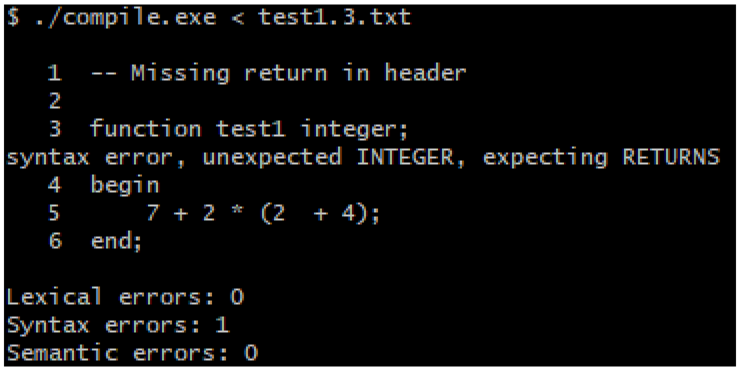


```
$ ./compile.exe < test1.2.txt

   1  -- Missing identifier in header
   2
   3  function returns integer;
syntax error, unexpected RETURNS, expecting IDENTIFIER
   4  begin
   5      7 + 2 * (2  + 4);
   6  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 5. Test case #5. Missing identifier in function header.

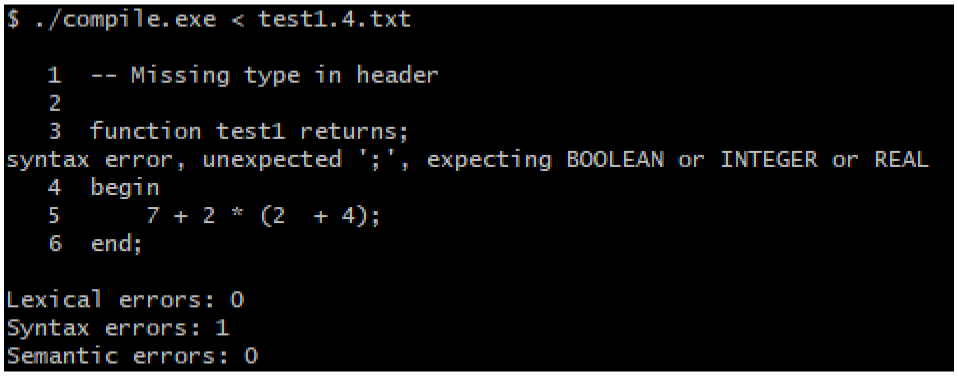


```
$ ./compile.exe < test1.3.txt

   1  -- Missing return in header
   2
   3  function test1 integer;
syntax error, unexpected INTEGER, expecting RETURNS
   4  begin
   5      7 + 2 * (2  + 4);
   6  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 6. Test case #6. Missing return keyword in function header.

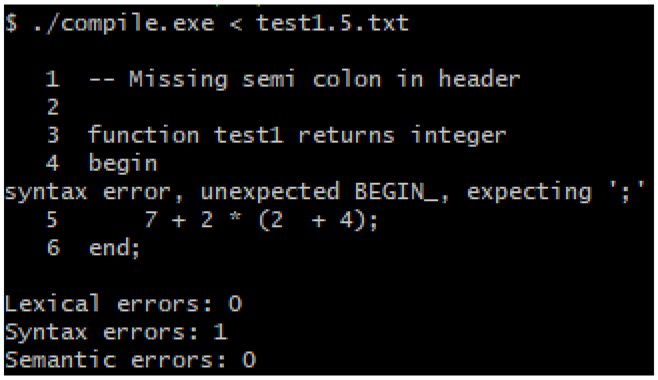


```
$ ./compile.exe < test1.4.txt

   1  -- Missing type in header
   2
   3  function test1 returns;
syntax error, unexpected ';', expecting BOOLEAN or INTEGER or REAL
   4  begin
   5      7 + 2 * (2  + 4);
   6  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 7. Test case #7. Missing type in function header.

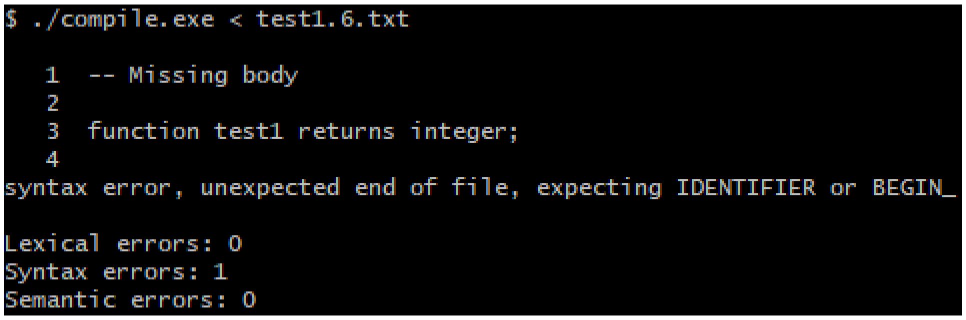


```
$ ./compile.exe < test1.5.txt

   1  -- Missing semi colon in header
   2
   3  function test1 returns integer
   4  begin
syntax error, unexpected BEGIN_, expecting ';'
   5      7 + 2 * (2  + 4);
   6  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 8. Test case #8. Missing semi colon in function header

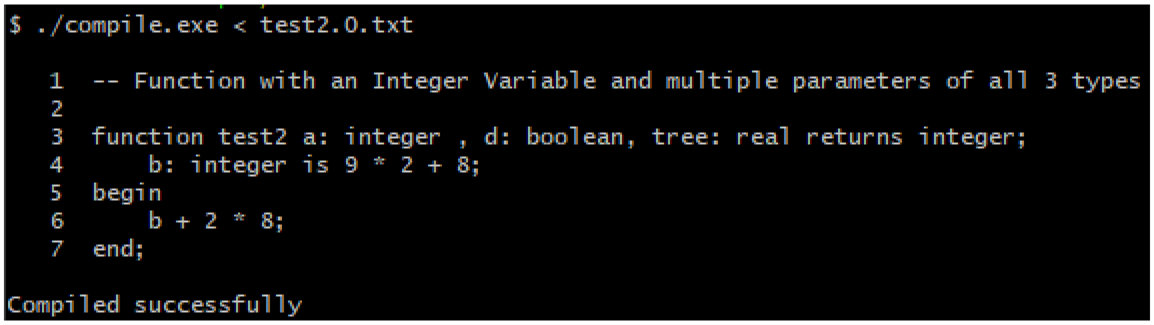


```
$ ./compile.exe < test1.6.txt

   1  -- Missing body
   2
   3  function test1 returns integer;
   4
syntax error, unexpected end of file, expecting IDENTIFIER or BEGIN_

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 9. Test case #9. Missing function body.

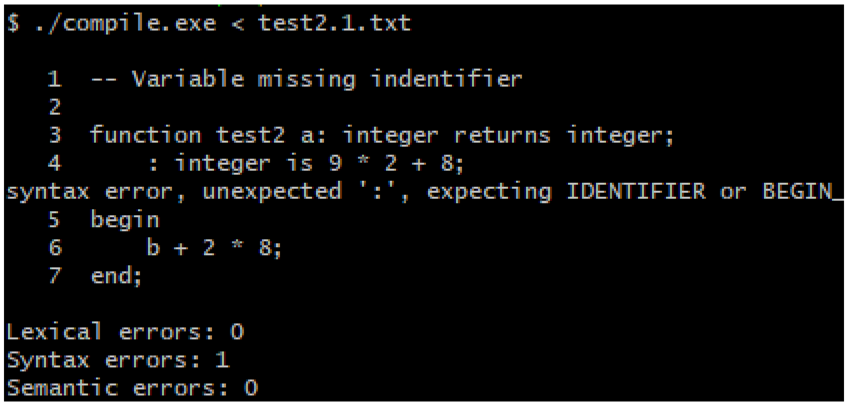


```
$ ./compile.exe < test2.0.txt

   1  -- Function with an Integer Variable and multiple parameters of all 3 types
   2
   3  function test2 a: integer , d: boolean, tree: real returns integer;
   4      b: integer is 9 * 2 + 8;
   5  begin
   6      b + 2 * 8;
   7  end;

Compiled successfully
```

Figure 10. Test case #10. Valid function with variable and multiple parameters.

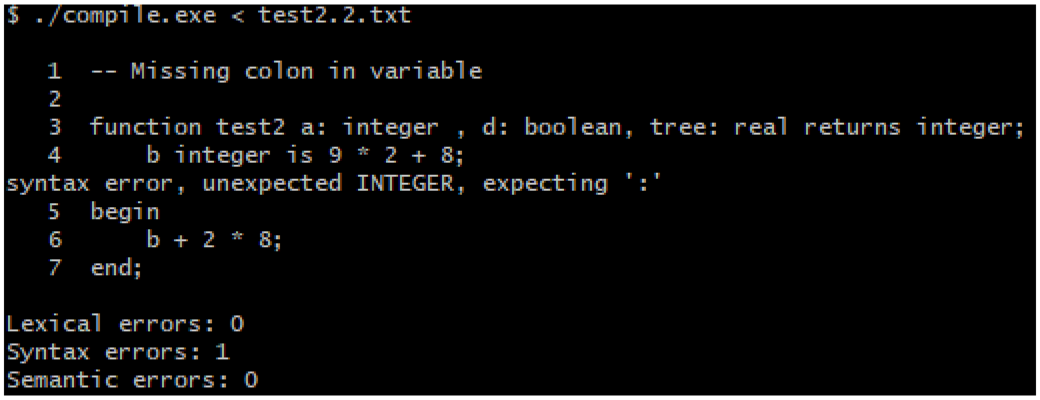


```
$ ./compile.exe < test2.1.txt

   1  -- Variable missing indentifier
   2
   3  function test2 a: integer returns integer;
   4      : integer is 9 * 2 + 8;
syntax error, unexpected ':', expecting IDENTIFIER or BEGIN_
   5  begin
   6      b + 2 * 8;
   7  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 11. Test case #11. Missing identifier in variable.

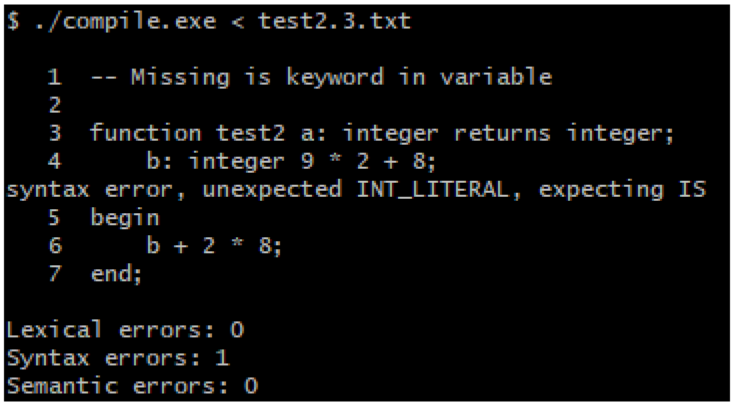


```
$ ./compile.exe < test2.2.txt

   1  -- Missing colon in variable
   2
   3  function test2 a: integer , d: boolean, tree: real returns integer;
   4      b integer is 9 * 2 + 8;
syntax error, unexpected INTEGER, expecting ':'
   5  begin
   6      b + 2 * 8;
   7  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 12. Test case #12. Missing colon in variable.

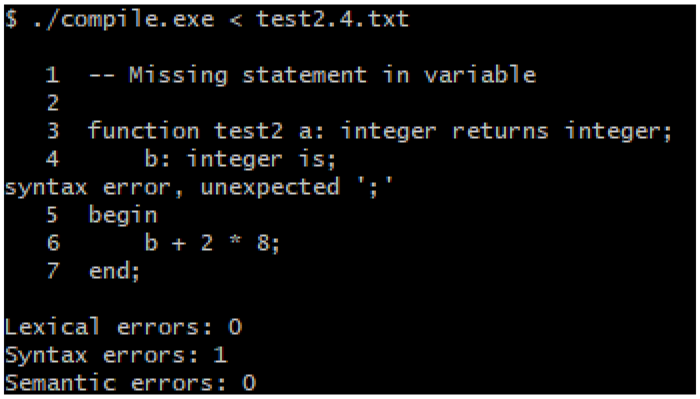


```
$ ./compile.exe < test2.3.txt

   1  -- Missing is keyword in variable
   2
   3  function test2 a: integer returns integer;
   4      b: integer 9 * 2 + 8;
syntax error, unexpected INT_LITERAL, expecting IS
   5  begin
   6      b + 2 * 8;
   7  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 13. Test case #13. Missing is keyword in variable.

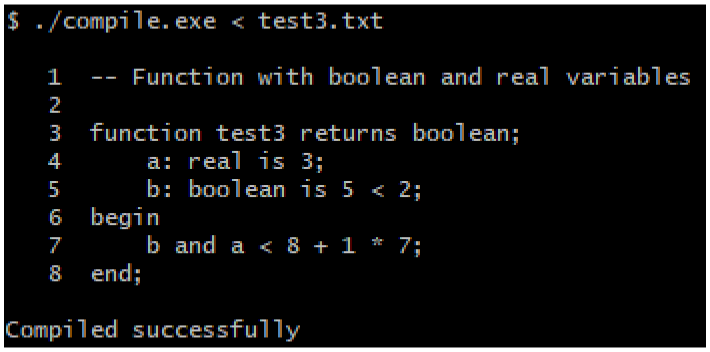


```
$ ./compile.exe < test2.4.txt

   1  -- Missing statement in variable
   2
   3  function test2 a: integer returns integer;
   4      b: integer is;
syntax error, unexpected ';'
   5  begin
   6      b + 2 * 8;
   7  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 14. Test case #14. Missing statement in variable.

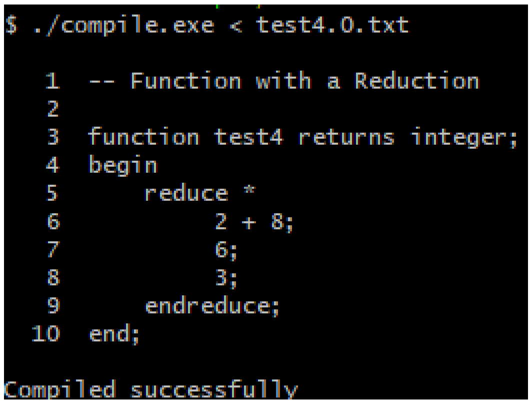


```
$ ./compile.exe < test3.txt

   1  -- Function with boolean and real variables
   2
   3  function test3 returns boolean;
   4      a: real is 3;
   5      b: boolean is 5 < 2;
   6  begin
   7      b and a < 8 + 1 * 7;
   8  end;

Compiled successfully
```

Figure 15. Test case #15. Boolean and real variables.

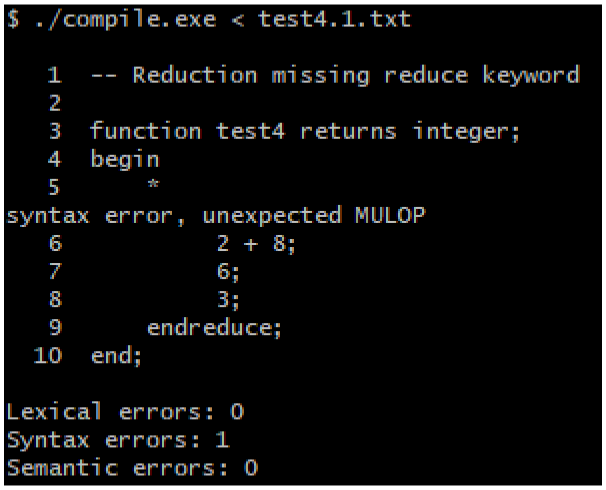


```
$ ./compile.exe < test4.0.txt

   1  -- Function with a Reduction
   2
   3  function test4 returns integer;
   4  begin
   5      reduce *
   6           2 + 8;
   7           6;
   8           3;
   9      endreduce;
  10  end;

Compiled successfully
```

Figure 16. Test case #16. Valid function with reduction.

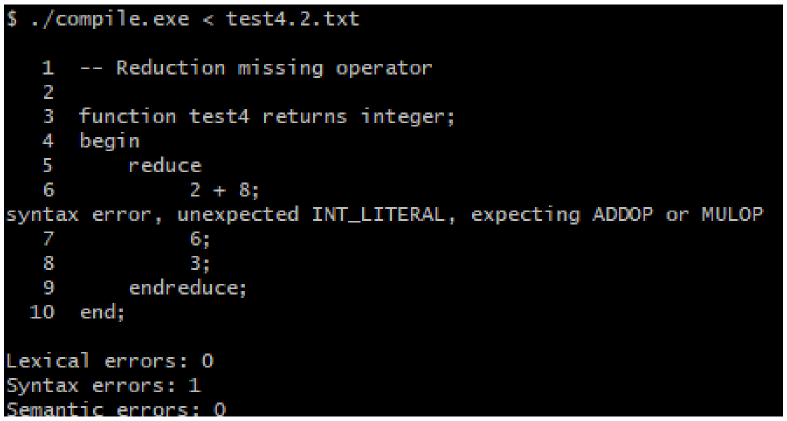


```
$ ./compile.exe < test4.1.txt

   1  -- Reduction missing reduce keyword
   2
   3  function test4 returns integer;
   4  begin
   5      *
syntax error, unexpected MULOP
   6           2 + 8;
   7           6;
   8           3;
   9      endreduce;
  10  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 17. Test case #17. Reduction missing reduce keyword.

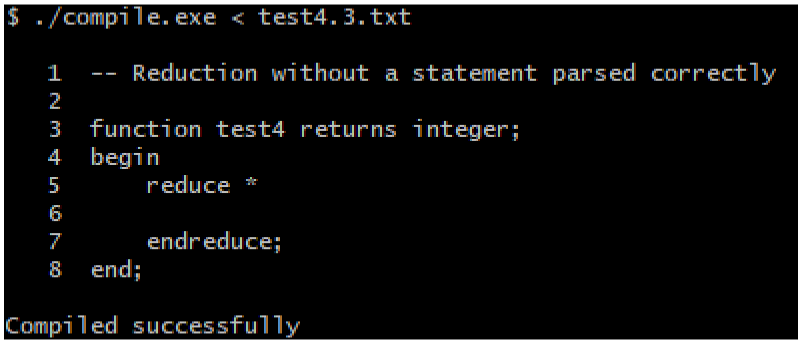


```
$ ./compile.exe < test4.2.txt

   1  -- Reduction missing operator
   2
   3  function test4 returns integer;
   4  begin
   5      reduce
   6           2 + 8;
syntax error, unexpected INT_LITERAL, expecting ADDOP or MULOP
   7           6;
   8           3;
   9      endreduce;
  10  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 18. Test case #18. Reduction missing operator.

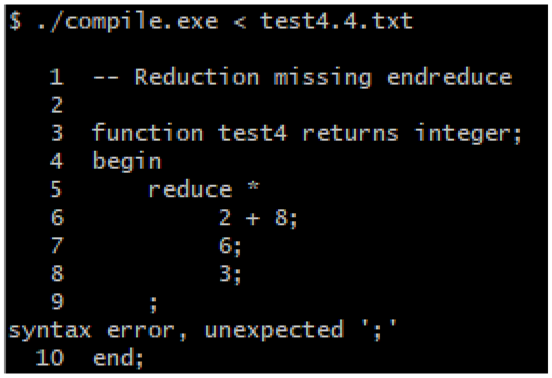


```
$ ./compile.exe < test4.3.txt

   1  -- Reduction without a statement parsed correctly
   2
   3  function test4 returns integer;
   4  begin
   5      reduce *
   6
   7      endreduce;
   8  end;

Compiled successfully
```

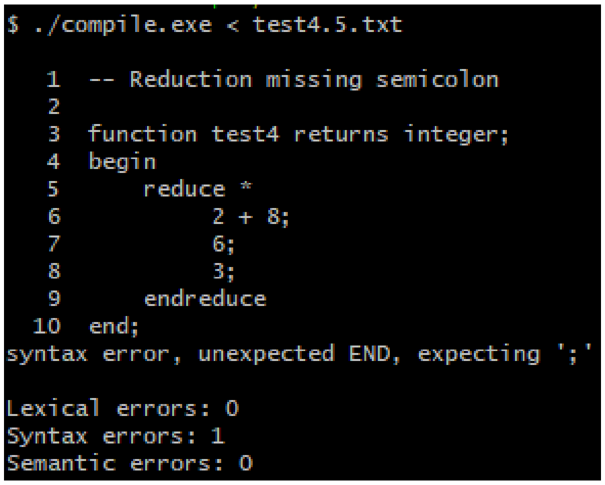
Figure 19. Test case #19. Reduction without a statement parsed correctly.



```
$ ./compile.exe < test4.4.txt

   1  -- Reduction missing endreduce
   2
   3  function test4 returns integer;
   4  begin
   5      reduce *
   6           2 + 8;
   7           6;
   8           3;
   9      ;
syntax error, unexpected ';'
  10  end;
```

Figure 20. Test case #20. Reduction without endreduce.

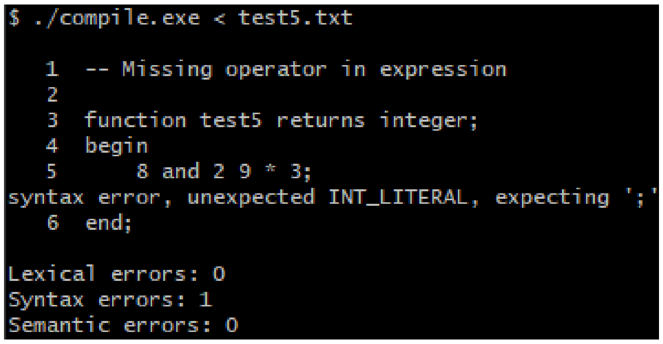


```
$ ./compile.exe < test4.5.txt

   1  -- Reduction missing semicolon
   2
   3  function test4 returns integer;
   4  begin
   5      reduce *
   6           2 + 8;
   7           6;
   8           3;
   9      endreduce
  10  end;
syntax error, unexpected END, expecting ';'

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 21. Test case #21. Reduction without semicolon.

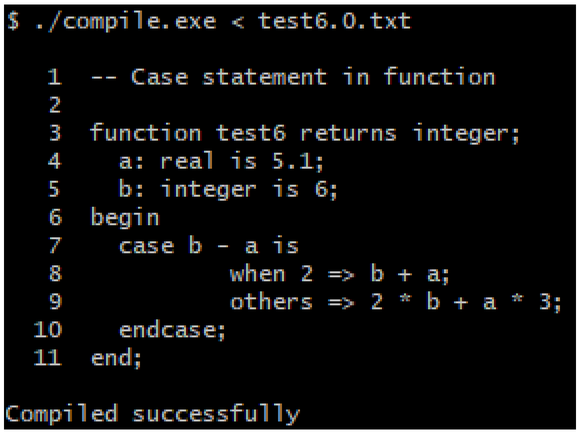


```
$ ./compile.exe < test5.txt

   1  -- Missing operator in expression
   2
   3  function test5 returns integer;
   4  begin
   5      8 and 2 9 * 3;
syntax error, unexpected INT_LITERAL, expecting ';'
   6  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 22. Test case #22. Missing operator in expression.

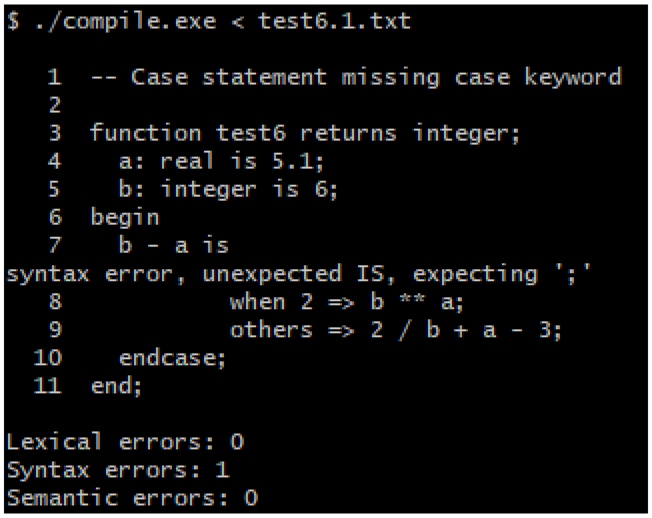


```
$ ./compile.exe < test6.0.txt

   1  -- Case statement in function
   2
   3  function test6 returns integer;
   4    a: real is 5.1;
   5    b: integer is 6;
   6  begin
   7    case b - a is
   8            when 2 => b + a;
   9            others => 2 * b + a * 3;
  10    endcase;
  11  end;

Compiled successfully
```

Figure 23. Test case #23. Successful parse of a valid function with a case statement.

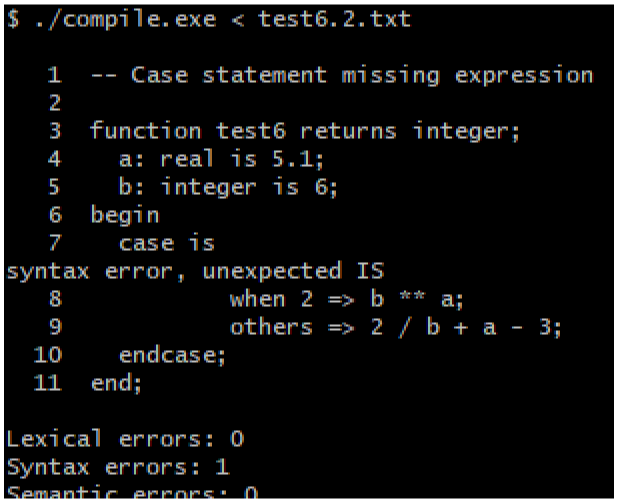


```
$ ./compile.exe < test6.1.txt

   1  -- Case statement missing case keyword
   2
   3  function test6 returns integer;
   4    a: real is 5.1;
   5    b: integer is 6;
   6  begin
   7    b - a is
syntax error, unexpected IS, expecting ';'
   8            when 2 => b ** a;
   9            others => 2 / b + a - 3;
  10    endcase;
  11  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 24. Test case #24. Case statement missing case keyword.



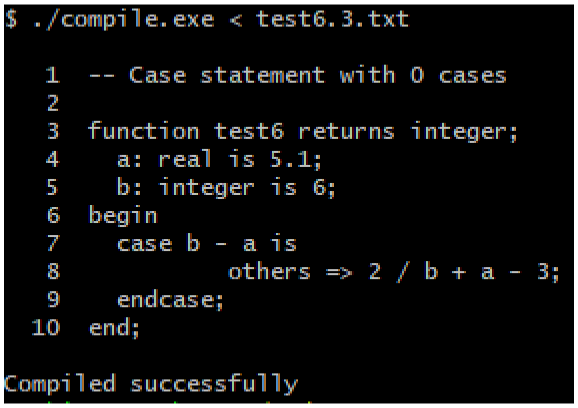
```
$ ./compile.exe < test6.2.txt

-- Case statement missing expression

function test6 returns integer;
  a: real is 5.1;
  b: integer is 6;
begin
  case is
syntax error, unexpected IS
          when 2 => b ** a;
          others => 2 / b + a - 3;
  endcase;
end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 25. Test case #25. Case statement missing expression.



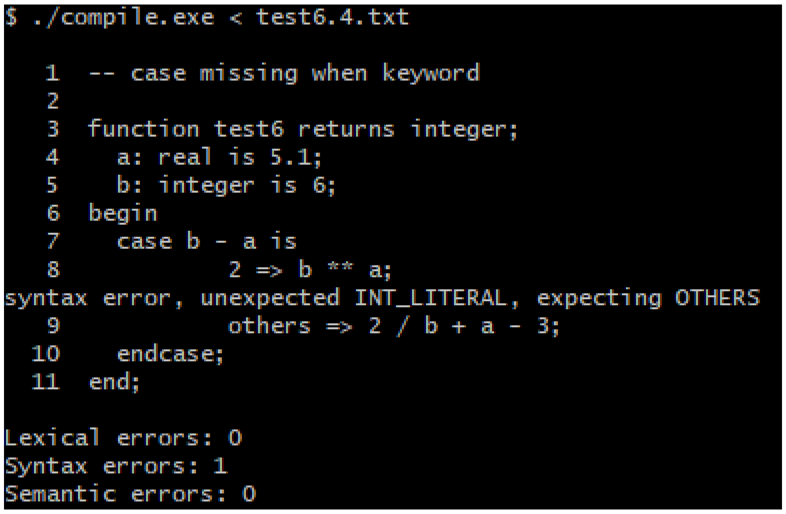
```
$ ./compile.exe < test6.3.txt

-- Case statement with 0 cases

function test6 returns integer;
  a: real is 5.1;
  b: integer is 6;
begin
  case b - a is
          others => 2 / b + a - 3;
  endcase;
end;

Compiled successfully
```

Figure 26. Test case #26. Case statement without any cases.



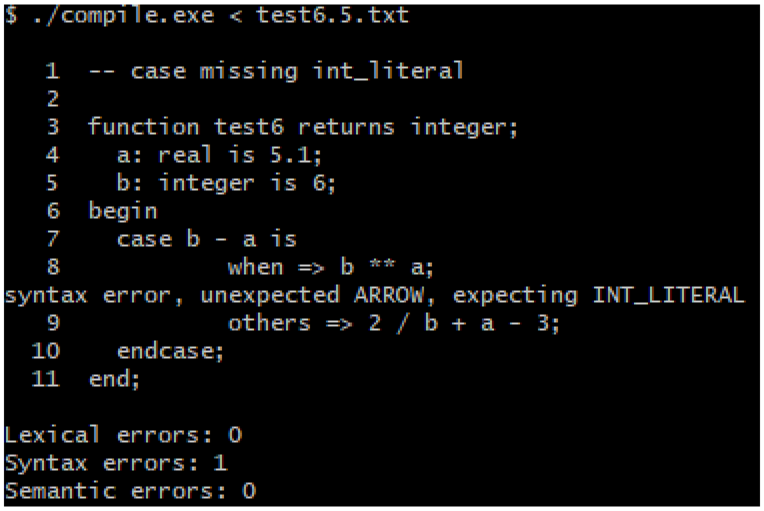
```
$ ./compile.exe < test6.4.txt

-- case missing when keyword

function test6 returns integer;
  a: real is 5.1;
  b: integer is 6;
begin
  case b - a is
          2 => b ** a;
syntax error, unexpected INT_LITERAL, expecting OTHERS
          others => 2 / b + a - 3;
  endcase;
end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 27. Test case #27. Case without when keyword.

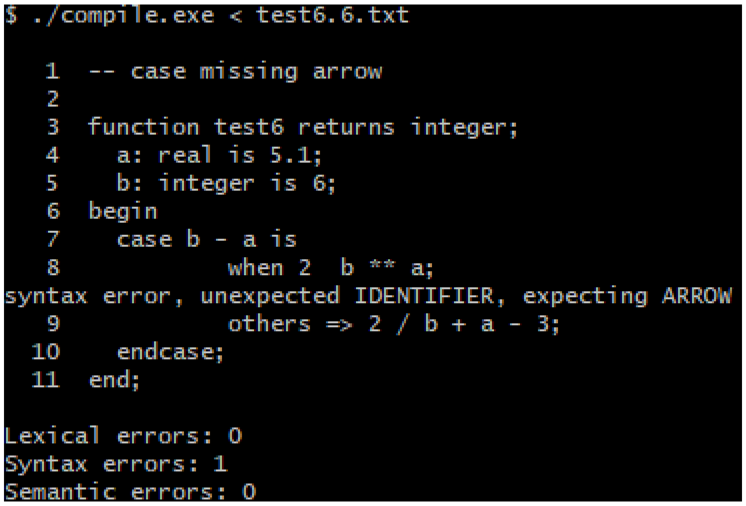


```
$ ./compile.exe < test6.5.txt

   1  -- case missing int_literal
   2
   3  function test6 returns integer;
   4    a: real is 5.1;
   5    b: integer is 6;
   6  begin
   7    case b - a is
   8            when => b ** a;
syntax error, unexpected ARROW, expecting INT_LITERAL
   9            others => 2 / b + a - 3;
  10    endcase;
  11  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 28. Test case #28. Case without int_literal.

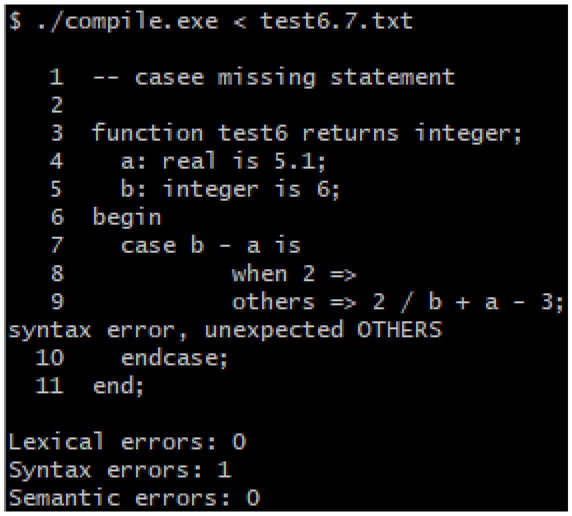


```
$ ./compile.exe < test6.6.txt

   1  -- case missing arrow
   2
   3  function test6 returns integer;
   4    a: real is 5.1;
   5    b: integer is 6;
   6  begin
   7    case b - a is
   8            when 2  b ** a;
syntax error, unexpected IDENTIFIER, expecting ARROW
   9            others => 2 / b + a - 3;
  10    endcase;
  11  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 29. Test case #29. Case without arrow.

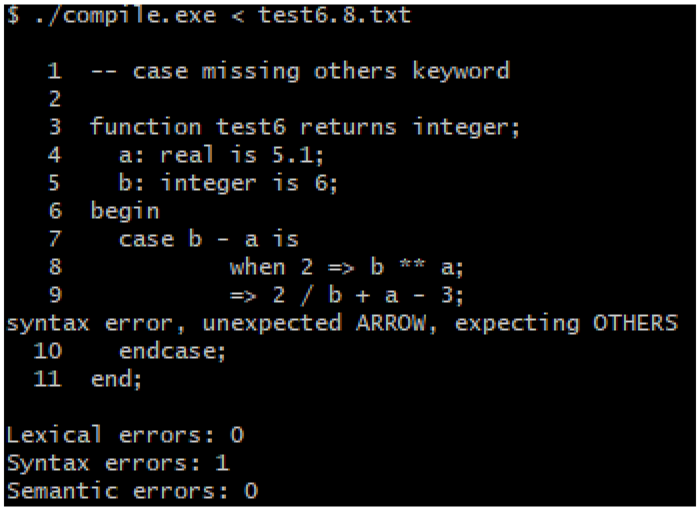


```
$ ./compile.exe < test6.7.txt

   1  -- casee missing statement
   2
   3  function test6 returns integer;
   4    a: real is 5.1;
   5    b: integer is 6;
   6  begin
   7    case b - a is
   8            when 2 =>
   9            others => 2 / b + a - 3;
syntax error, unexpected OTHERS
  10    endcase;
  11  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 30. Test case #30. Case without statement.

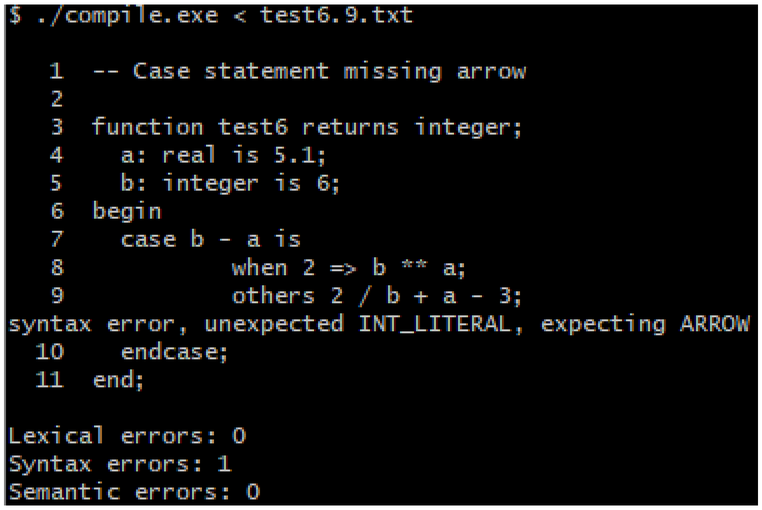


```
$ ./compile.exe < test6.8.txt

   1  -- case missing others keyword
   2
   3  function test6 returns integer;
   4    a: real is 5.1;
   5    b: integer is 6;
   6  begin
   7    case b - a is
   8            when 2 => b ** a;
   9            => 2 / b + a - 3;
syntax error, unexpected ARROW, expecting OTHERS
  10    endcase;
  11  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 31. Test case #31. Case without others keyword.

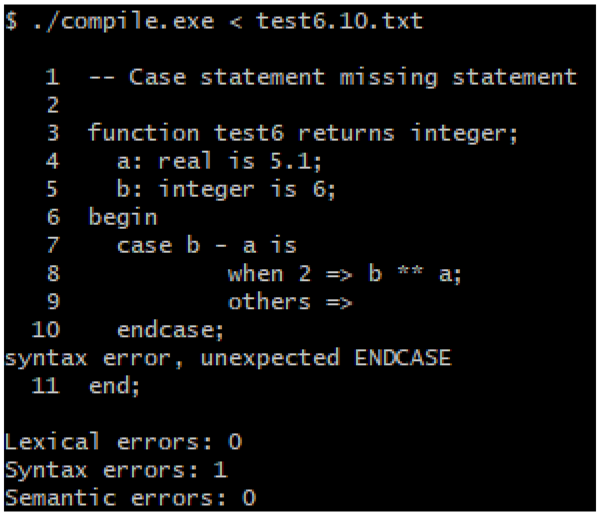


```
$ ./compile.exe < test6.9.txt

   1  -- Case statement missing arrow
   2
   3  function test6 returns integer;
   4    a: real is 5.1;
   5    b: integer is 6;
   6  begin
   7    case b - a is
   8            when 2 => b ** a;
   9            others 2 / b + a - 3;
syntax error, unexpected INT_LITERAL, expecting ARROW
  10    endcase;
  11  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 32. Test case #32. Case statement without arrow.

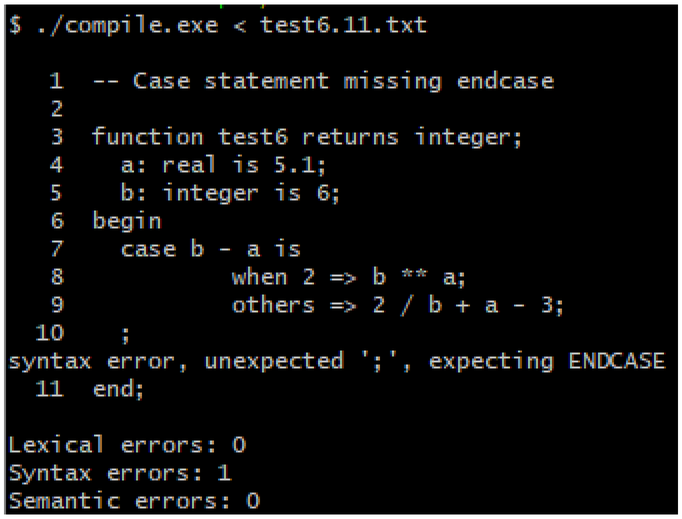


```
$ ./compile.exe < test6.10.txt

   1  -- Case statement missing statement
   2
   3  function test6 returns integer;
   4    a: real is 5.1;
   5    b: integer is 6;
   6  begin
   7    case b - a is
   8            when 2 => b ** a;
   9            others =>
  10    endcase;
syntax error, unexpected ENDCASE
  11  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 33. Test case #33. Case statement without statement.

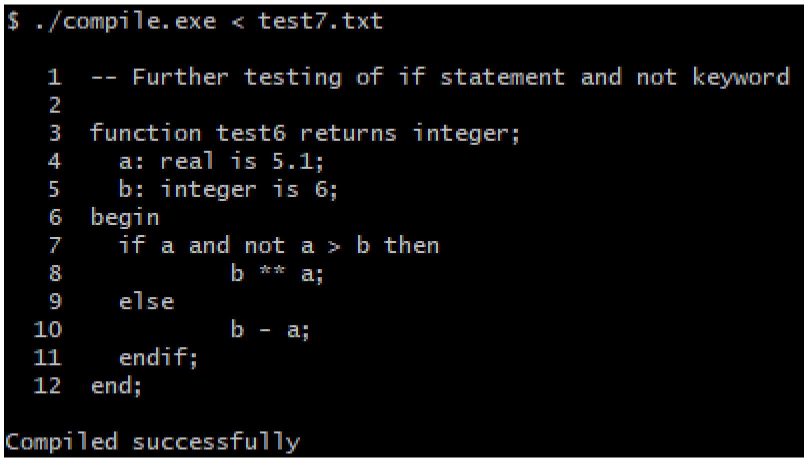


```
$ ./compile.exe < test6.11.txt

   1  -- Case statement missing endcase
   2
   3  function test6 returns integer;
   4    a: real is 5.1;
   5    b: integer is 6;
   6  begin
   7    case b - a is
   8            when 2 => b ** a;
   9            others => 2 / b + a - 3;
  10    ;
syntax error, unexpected ';', expecting ENDCASE
  11  end;

Lexical errors: 0
Syntax errors: 1
Semantic errors: 0
```

Figure 34. Test case #34. Case statement without endcase.



```
$ ./compile.exe < test7.txt

   1  -- Further testing of if statement and not keyword
   2
   3  function test6 returns integer;
   4    a: real is 5.1;
   5    b: integer is 6;
   6  begin
   7    if a and not a > b then
   8            b ** a;
   9    else
  10            b - a;
  11    endif;
  12  end;

Compiled successfully
```

Figure 35. Test case #35. Further testing of if statement and other keywords.