Project Approach

While trying to understand how to write and what to write for our week 5 discussion post, I ended up going ahead and starting on project three. As a lot of the course content for week five and week six went hand in hand. To be able to understand the some of these concepts, I thought working with project three for a hands-on approach would allow me to get a better understanding of what week five's discussion post was asking. It also allowed me to get a head start of project three.

Reviewing the content for week six gave me a better grasp of how to construct the interpreter, and how the semantic rules are structured in the bison file (parser.y). I think the videos in the content for week six is where I got a lot of help for project three, which ultimately helped me complete the discussion post for week five. While following along with the course content and applying the concepts to project three, made me really want to finish project three before writing my discussion post (at least finish enough to compile my program and receive a calculated result). Once I got project three able to read and compiled a program that met discussion five's instructions, I started further work on project three.

I ran into a few issues (will discuss further in the lessons learned section) that required a little more time and research from outside sources. However, for the most part I was able to get most of the information from our course content, mainly in week five and six.

Test Cases

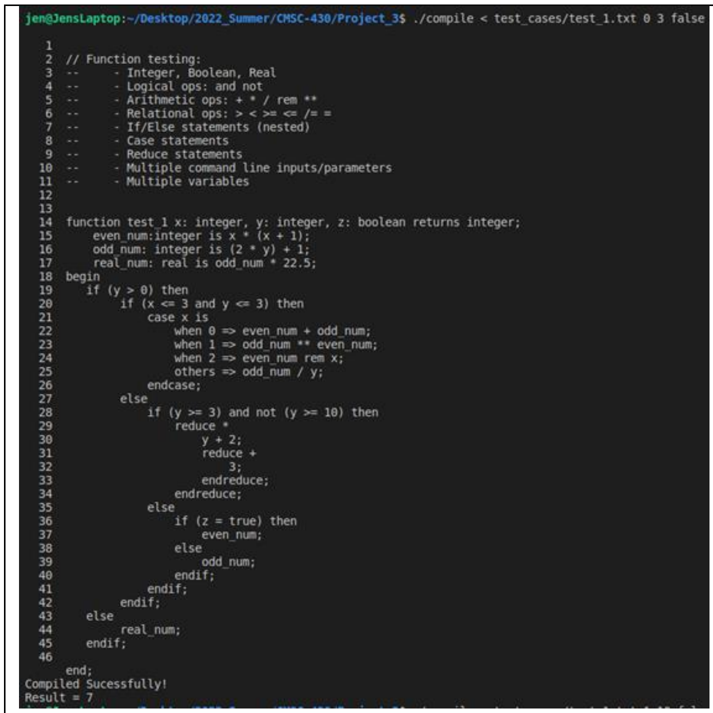
Test Case 1:

- Integer, Boolean, Real
- Logical ops: and not
- Arithmetic ops: + * / rem **
- Relational ops: > < >= <= /= =
- If/Else statements (nested)
- Case statements
- Reduce statements
- Multiple command line inputs/parameters
- Multiple variables

Test Num.	x	y	z	Result
Test 1	0	3	false	7
Test 2	1	2	false	25
Test 3	2	2	false	0
Test 4	3	1	false	3
Test 5	4	4	false	18
Test 6	5	11	true	30
Test 7	5	11	false	23
Test 8	6	0	true	22.5

Screen Shots for Test Case 1:

- Test 1



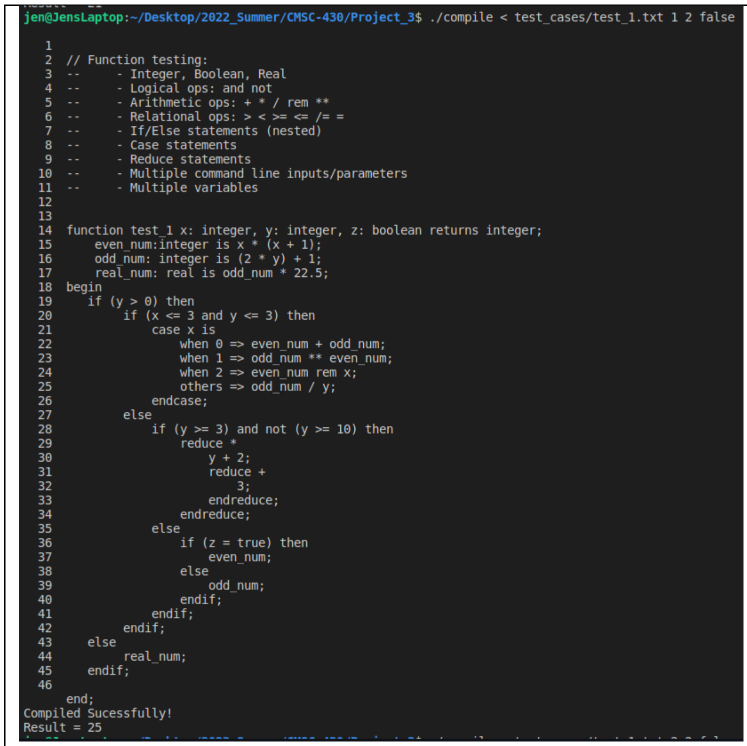
```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_1.txt 0 3 false

   1
   2  // Function testing:
   3  --      - Integer, Boolean, Real
   4  --      - Logical ops: and not
   5  --      - Arithmetic ops: + * / rem **
   6  --      - Relational ops: > < >= <= /= =
   7  --      - If/Else statements (nested)
   8  --      - Case statements
   9  --      - Reduce statements
  10  --      - Multiple command line inputs/parameters
  11  --      - Multiple variables
  12
  13
  14  function test_1 x: integer, y: integer, z: boolean returns integer;
  15      even_num:integer is x * (x + 1);
  16      odd_num: integer is (2 * y) + 1;
  17      real_num: real is odd_num * 22.5;
  18  begin
  19     if (y > 0) then
  20          if (x <= 3 and y <= 3) then
  21              case x is
  22                  when 0 => even_num + odd_num;
  23                  when 1 => odd_num ** even_num;
  24                  when 2 => even_num rem x;
  25                  others => odd_num / y;
  26              endcase;
  27          else
  28              if (y >= 3) and not (y >= 10) then
  29                  reduce *
  30                      y + 2;
  31                      reduce +
  32                          3;
  33                      endreduce;
  34                  endreduce;
  35              else
  36                  if (z = true) then
  37                      even_num;
  38                  else
  39                      odd_num;
  40                  endif;
  41              endif;
  42          endif;
  43     else
  44          real_num;
  45     endif;
  46
      end;
Compiled Sucessfully!
Result = 7
```

(Passing parameters x=0, y=3, z=false) This test should enter the first two if conditions and then move into the case statement. Where x=0, and returns the 'when 0' case (Line 22). We have:

even_num + odd_num

(0 * (0 + 1)) + ((2 * 3) + 1) = 7

- Test 2



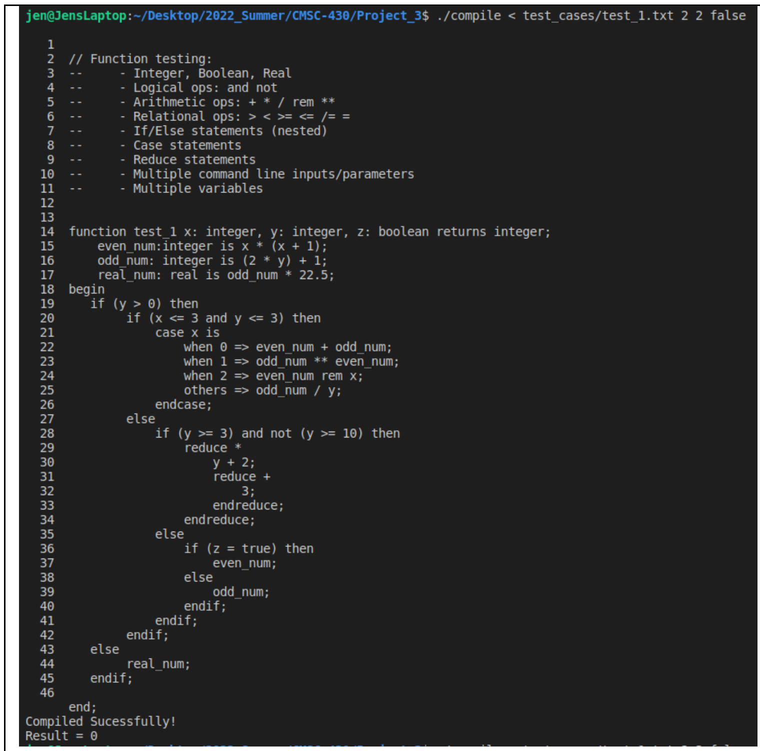
```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_1.txt 1 2 false

   1
   2  // Function testing:
   3  --     - Integer, Boolean, Real
   4  --     - Logical ops: and not
   5  --     - Arithmetic ops: + * / rem **
   6  --     - Relational ops: > < >= <= /= =
   7  --     - If/Else statements (nested)
   8  --     - Case statements
   9  --     - Reduce statements
  10  --     - Multiple command line inputs/parameters
  11  --     - Multiple variables
  12
  13
  14  function test_1 x: integer, y: integer, z: boolean returns integer;
  15      even_num:integer is x * (x + 1);
  16      odd_num: integer is (2 * y) + 1;
  17      real_num: real is odd_num * 22.5;
  18  begin
  19     if (y > 0) then
  20          if (x <= 3 and y <= 3) then
  21              case x is
  22                  when 0 => even_num + odd_num;
  23                  when 1 => odd_num ** even_num;
  24                  when 2 => even_num rem x;
  25                  others => odd_num / y;
  26              endcase;
  27          else
  28              if (y >= 3) and not (y >= 10) then
  29                  reduce *
  30                      y + 2;
  31                      reduce +
  32                          3;
  33                      endreduce;
  34                  endreduce;
  35              else
  36                  if (z = true) then
  37                      even_num;
  38                  else
  39                      odd_num;
  40                  endif;
  41              endif;
  42          endif;
  43     else
  44          real_num;
  45     endif;
  46

      end;
Compiled Sucessfully!
Result = 25
```

(Passing parameters x=1, y=2, z=false) This test should enter the first two if conditions and then move into the case statement. Where x=1, and returns the 'when 1' case (Line 23). We have:

odd_num ** even_num

((2 * 2) + 1) ** ((1 + (1 * 1) = 25

- Test 3



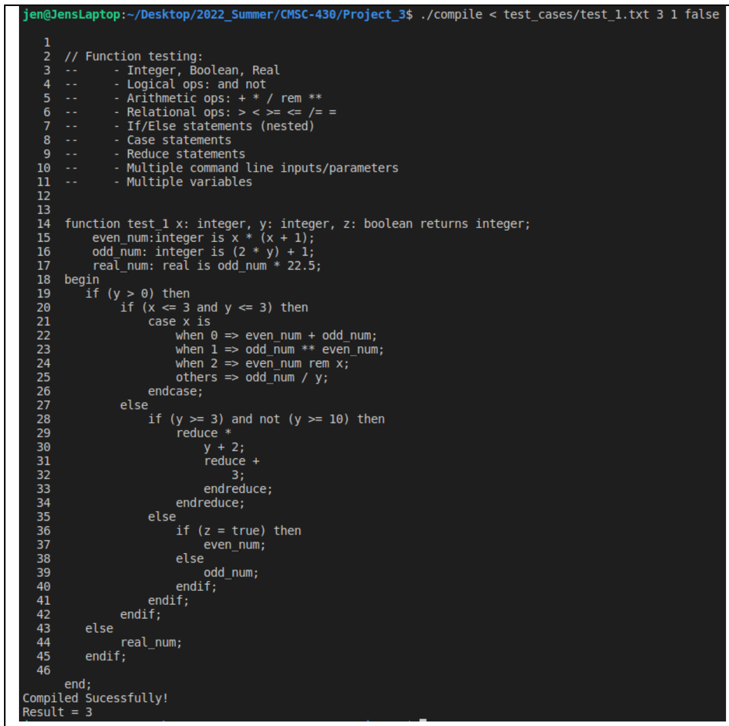
```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_1.txt 2 2 false

   1
   2  // Function testing:
   3  --     - Integer, Boolean, Real
   4  --     - Logical ops: and not
   5  --     - Arithmetic ops: + * / rem **
   6  --     - Relational ops: > < >= <= /= =
   7  --     - If/Else statements (nested)
   8  --     - Case statements
   9  --     - Reduce statements
  10  --     - Multiple command line inputs/parameters
  11  --     - Multiple variables
  12
  13
  14  function test_1 x: integer, y: integer, z: boolean returns integer;
  15      even_num:integer is x * (x + 1);
  16      odd_num: integer is (2 * y) + 1;
  17      real_num: real is odd_num * 22.5;
  18  begin
  19     if (y > 0) then
  20          if (x <= 3 and y <= 3) then
  21              case x is
  22                  when 0 => even_num + odd_num;
  23                  when 1 => odd_num ** even_num;
  24                  when 2 => even_num rem x;
  25                  others => odd_num / y;
  26              endcase;
  27          else
  28              if (y >= 3) and not (y >= 10) then
  29                  reduce *
  30                      y + 2;
  31                      reduce +
  32                          3;
  33                      endreduce;
  34                  endreduce;
  35              else
  36                  if (z = true) then
  37                      even_num;
  38                  else
  39                      odd_num;
  40                  endif;
  41              endif;
  42          endif;
  43     else
  44          real_num;
  45     endif;
  46

      end;
Compiled Sucessfully!
Result = 0
```

(Passing parameters x=2, y=2, z=false) This test should enter the first two if conditions and then move into the case statement. Where x=2, and returns the 'when 2' case (Line 24). We have:

even_num rem x

(1 + (1 * 1)) rem (2) = 0

- Test 4



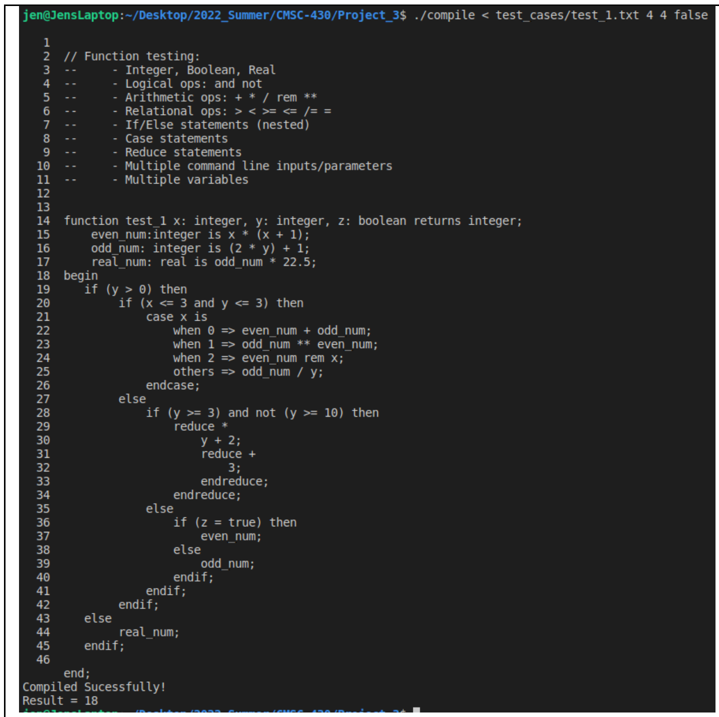
```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_1.txt 3 1 false

   1
   2  // Function testing:
   3  --      - Integer, Boolean, Real
   4  --      - Logical ops: and not
   5  --      - Arithmetic ops: + * / rem **
   6  --      - Relational ops: > < >= <= /= =
   7  --      - If/Else statements (nested)
   8  --      - Case statements
   9  --      - Reduce statements
  10  --      - Multiple command line inputs/parameters
  11  --      - Multiple variables
  12
  13
  14  function test_1 x: integer, y: integer, z: boolean returns integer;
  15      even_num:integer is x * (x + 1);
  16      odd_num: integer is (2 * y) + 1;
  17      real_num: real is odd_num * 22.5;
  18  begin
  19     if (y > 0) then
  20          if (x <= 3 and y <= 3) then
  21              case x is
  22                  when 0 => even_num + odd_num;
  23                  when 1 => odd_num ** even_num;
  24                  when 2 => even_num rem x;
  25                  others => odd_num / y;
  26              endcase;
  27          else
  28              if (y >= 3) and not (y >= 10) then
  29                  reduce *
  30                      y + 2;
  31                      reduce +
  32                          3;
  33                      endreduce;
  34                  endreduce;
  35              else
  36                  if (z = true) then
  37                      even_num;
  38                  else
  39                      odd_num;
  40                  endif;
  41              endif;
  42          endif;
  43     else
  44          real_num;
  45     endif;
  46
      end;
Compiled Sucessfully!
Result = 3
```

(Passing parameter x=3, y=1, z=false) This test should enter the first two if conditions and then move into the case statement. Where x=3, and returns the 'others' case (Line 25). We have:

odd_num / y

((2 * 1) + 1) / (1) = 3

- Test 5



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_1.txt 4 4 false

   1
   2  // Function testing:
   3  --      - Integer, Boolean, Real
   4  --      - Logical ops: and not
   5  --      - Arithmetic ops: + * / rem **
   6  --      - Relational ops: > < >= <= /= =
   7  --      - If/Else statements (nested)
   8  --      - Case statements
   9  --      - Reduce statements
  10  --      - Multiple command line inputs/parameters
  11  --      - Multiple variables
  12
  13
  14  function test_1 x: integer, y: integer, z: boolean returns integer;
  15      even_num:integer is x * (x + 1);
  16      odd_num: integer is (2 * y) + 1;
  17      real_num: real is odd_num * 22.5;
  18  begin
  19     if (y > 0) then
  20          if (x <= 3 and y <= 3) then
  21              case x is
  22                  when 0 => even_num + odd_num;
  23                  when 1 => odd_num ** even_num;
  24                  when 2 => even_num rem x;
  25                  others => odd_num / y;
  26              endcase;
  27          else
  28              if (y >= 3) and not (y >= 10) then
  29                  reduce *
  30                      y + 2;
  31                      reduce +
  32                          3;
  33                      endreduce;
  34                  endreduce;
  35              else
  36                  if (z = true) then
  37                      even_num;
  38                  else
  39                      odd_num;
  40                  endif;
  41              endif;
  42          endif;
  43     else
  44          real_num;
  45     endif;
  46
      end;
Compiled Sucessfully!
Result = 18
```

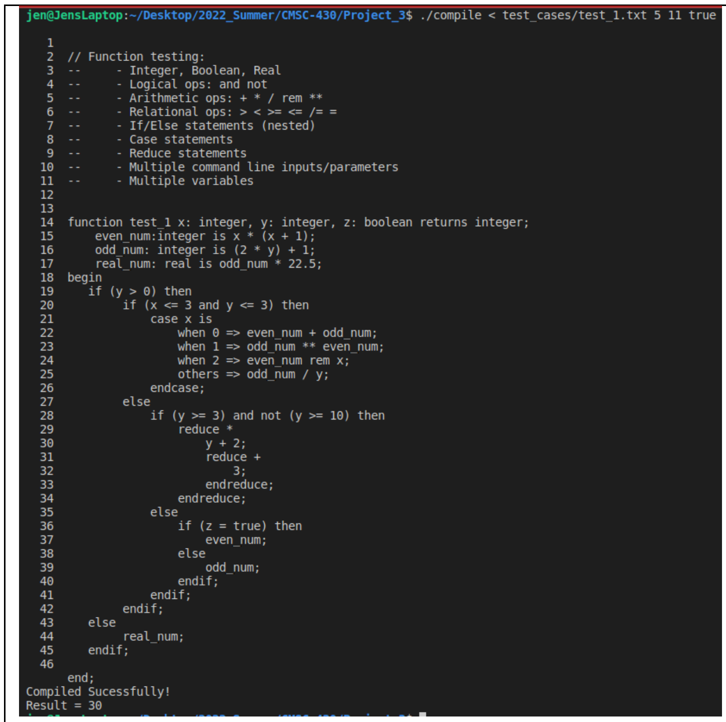
(Passing parameters x=4, y=4, z=false) This test should enter the first if condition, then move to the next if statement where it will hit the else condition. From there, it should go into the next if condition (Line 28), where it will then calculate the reduce statement (Line 29).

Reduce statement == 3 * 6 = 18

(4) +2 = 6

3

- Test 6

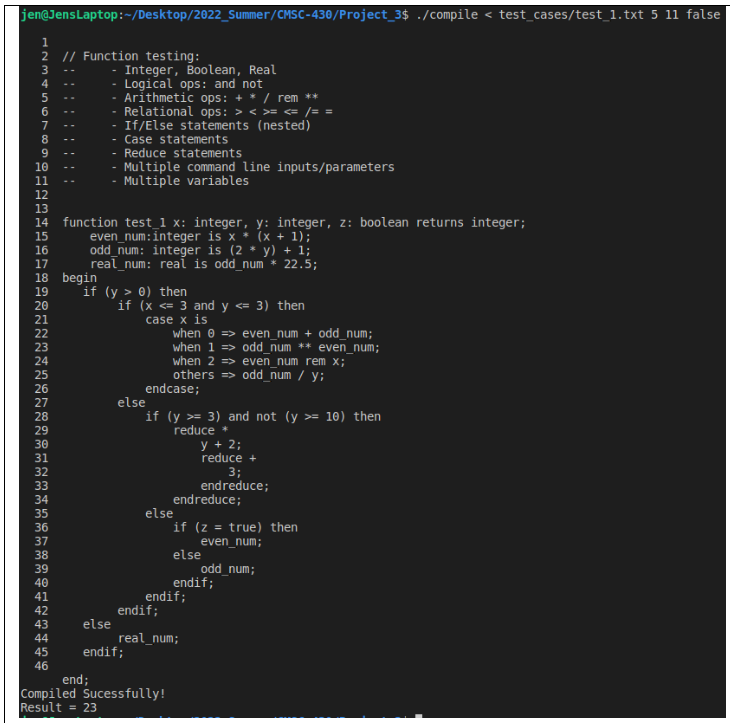


```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_1.txt 5 11 true

   1
   2  // Function testing:
   3  --      - Integer, Boolean, Real
   4  --      - Logical ops: and not
   5  --      - Arithmetic ops: + * / rem **
   6  --      - Relational ops: > < >= <= /= =
   7  --      - If/Else statements (nested)
   8  --      - Case statements
   9  --      - Reduce statements
  10  --      - Multiple command line inputs/parameters
  11  --      - Multiple variables
  12
  13
  14  function test_1 x: integer, y: integer, z: boolean returns integer;
  15      even_num:integer is x * (x + 1);
  16      odd_num: integer is (2 * y) + 1;
  17      real_num: real is odd_num * 22.5;
  18  begin
  19     if (y > 0) then
  20          if (x <= 3 and y <= 3) then
  21              case x is
  22                  when 0 => even_num + odd_num;
  23                  when 1 => odd_num ** even_num;
  24                  when 2 => even_num rem x;
  25                  others => odd_num / y;
  26              endcase;
  27          else
  28              if (y >= 3) and not (y >= 10) then
  29                  reduce *
  30                      y + 2;
  31                      reduce +
  32                          3;
  33                      endreduce;
  34                  endreduce;
  35              else
  36                  if (z = true) then
  37                      even_num;
  38                  else
  39                      odd_num;
  40                  endif;
  41              endif;
  42          endif;
  43     else
  44          real_num;
  45     endif;
  46
      end;
Compiled Sucessfully!
Result = 30
```

(Passing parameters x=5, y=11, z=true) This test should enter the first if condition, then move to the next if statement where it will hit the else condition. From there, it will move to the next else condition, then into the next if condition (Line 37). Where z is true, and will return even_num.

(5 * (5 + 1)) = 30

- Test 7

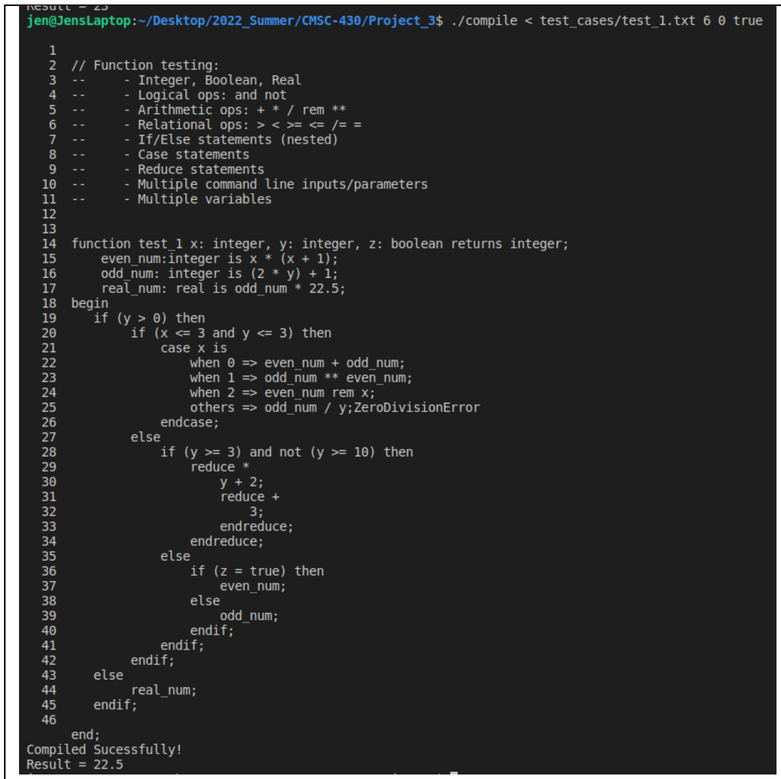


```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_1.txt 5 11 false

   1
   2  // Function testing:
   3  --      - Integer, Boolean, Real
   4  --      - Logical ops: and not
   5  --      - Arithmetic ops: + * / rem **
   6  --      - Relational ops: > < >= <= /= =
   7  --      - If/Else statements (nested)
   8  --      - Case statements
   9  --      - Reduce statements
  10  --      - Multiple command line inputs/parameters
  11  --      - Multiple variables
  12
  13
  14  function test_1 x: integer, y: integer, z: boolean returns integer;
  15      even_num:integer is x * (x + 1);
  16      odd_num: integer is (2 * y) + 1;
  17      real_num: real is odd_num * 22.5;
  18  begin
  19     if (y > 0) then
  20          if (x <= 3 and y <= 3) then
  21              case x is
  22                  when 0 => even_num + odd_num;
  23                  when 1 => odd_num ** even_num;
  24                  when 2 => even_num rem x;
  25                  others => odd_num / y;
  26              endcase;
  27          else
  28              if (y >= 3) and not (y >= 10) then
  29                  reduce *
  30                      y + 2;
  31                      reduce +
  32                          3;
  33                      endreduce;
  34                  endreduce;
  35              else
  36                  if (z = true) then
  37                      even_num;
  38                  else
  39                      odd_num;
  40                  endif;
  41              endif;
  42          endif;
  43     else
  44          real_num;
  45     endif;
  46
      end;
Compiled Sucessfully!
Result = 23
```

(Passing parameter x=5, y=11, z=false) This test should enter the first if condition, then move to the next if statement where it will hit the else condition. From there, it will move to the next else condition, then into the following else condition (Line 38). Where z is false, and will return odd_num.

((2 * 11) + 1) = 23

- Test 8



```
Result = 23
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_1.txt 6 0 true

   1
   2  // Function testing:
   3  --      - Integer, Boolean, Real
   4  --      - Logical ops: and not
   5  --      - Arithmetic ops: + * / rem **
   6  --      - Relational ops: > < >= <= /= =
   7  --      - If/Else statements (nested)
   8  --      - Case statements
   9  --      - Reduce statements
  10  --      - Multiple command line inputs/parameters
  11  --      - Multiple variables
  12
  13
  14  function test_1 x: integer, y: integer, z: boolean returns integer;
  15      even_num:integer is x * (x + 1);
  16      odd_num: integer is (2 * y) + 1;
  17      real_num: real is odd_num * 22.5;
  18  begin
  19     if (y > 0) then
  20          if (x <= 3 and y <= 3) then
  21              case x is
  22                  when 0 => even_num + odd_num;
  23                  when 1 => odd_num ** even_num;
  24                  when 2 => even_num rem x;
  25                  others => odd_num / y;ZeroDivisionError
  26              endcase;
  27          else
  28              if (y >= 3) and not (y >= 10) then
  29                  reduce *
  30                      y + 2;
  31                      reduce +
  32                          3;
  33                      endreduce;
  34                  endreduce;
  35              else
  36                  if (z = true) then
  37                      even_num;
  38                  else
  39                      odd_num;
  40                  endif;
  41              endif;
  42          endif;
  43     else
  44          real_num;
  45     endif;
  46
      end;
Compiled Sucessfully!
Result = 22.5
```

(Passing parameters x=6, y=0, z=true) This test should enter the first else condition (Line 43). As, y is zero and it will return real_num.

Odd_num * 22.5

((2 * 0) + 1) * 22.5 = 22.5

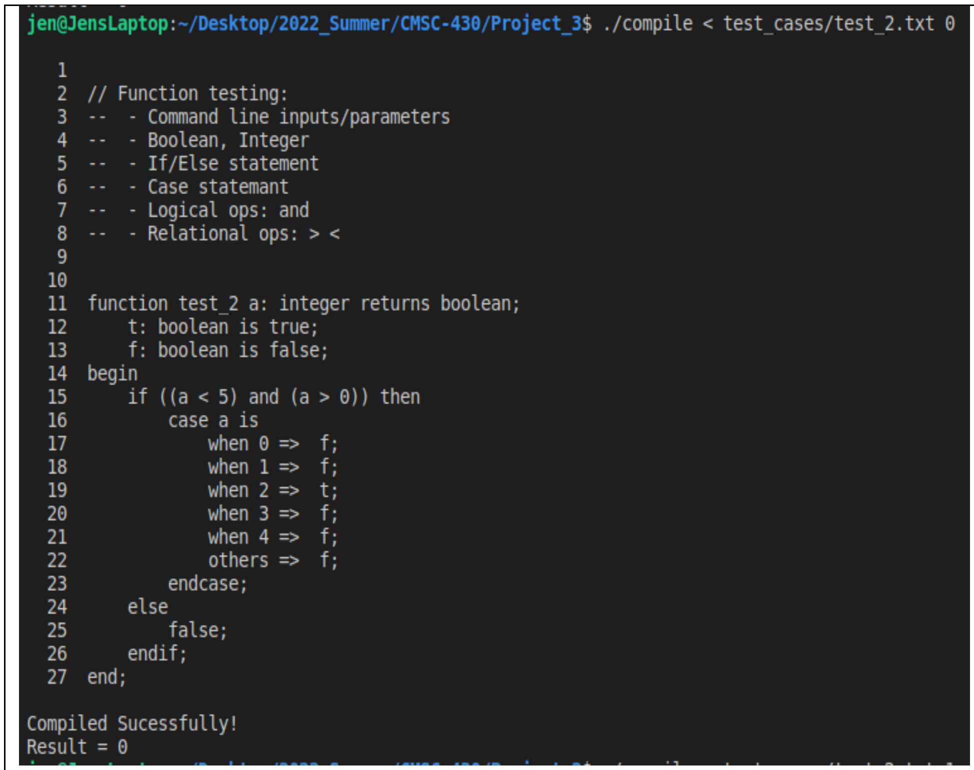
Test Case 2:

- Integer, Boolean
- Logical ops: and
- Relational ops: > <
- If/Else statements
- Case statements
- Command line inputs/parameters
- Multiple variables

Test Num.	a	Result
Test 1	0	0
Test 2	1	0
Test 3	2	1
Test 4	3	0
Test 5	4	0
Test 6	5	0
Test 7	6	0

Test Case 2 Screen Shots:

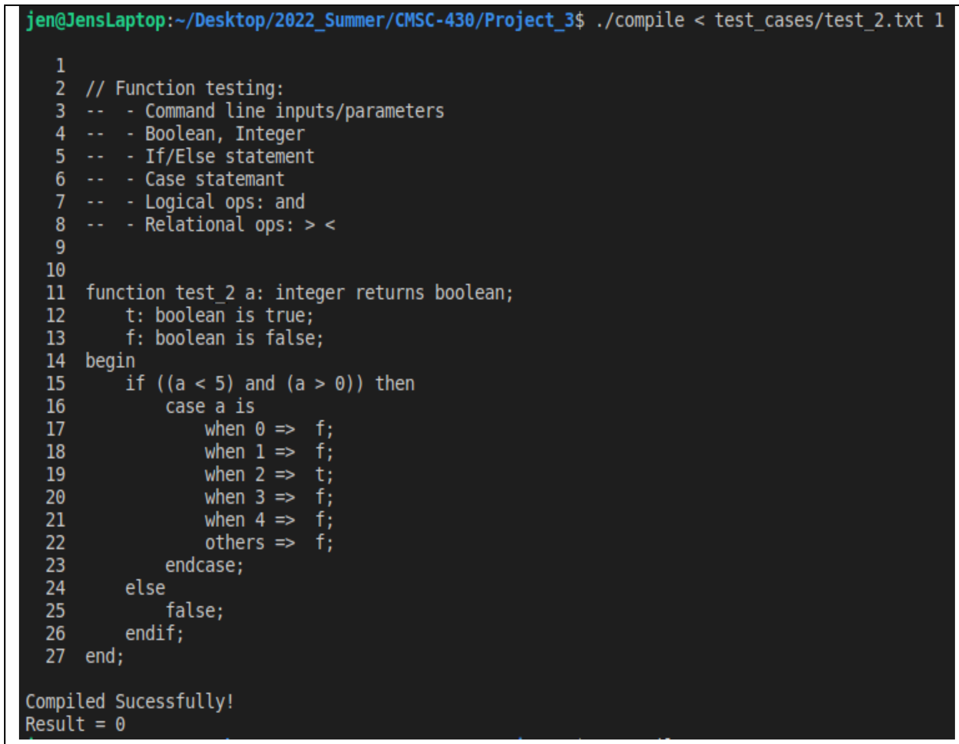
- Test 1



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_2.txt 0

   1
   2  // Function testing:
   3  --  - Command line inputs/parameters
   4  --  - Boolean, Integer
   5  --  - If/Else statement
   6  --  - Case statemant
   7  --  - Logical ops: and
   8  --  - Relational ops: > <
   9
  10
  11  function test_2 a: integer returns boolean;
  12      t: boolean is true;
  13      f: boolean is false;
  14  begin
  15      if ((a < 5) and (a > 0)) then
  16          case a is
  17              when 0 =>  f;
  18              when 1 =>  f;
  19              when 2 =>  t;
  20              when 3 =>  f;
  21              when 4 =>  f;
  22              others =>  f;
  23          endcase;
  24      else
  25          false;
  26      endif;
  27  end;

Compiled Sucessfully!
Result = 0
```

(Passing parameter a=0) This test should enter the if condition, then enter the case condition. Where a meets 'when 0' (Line 17) will return f (f=false).

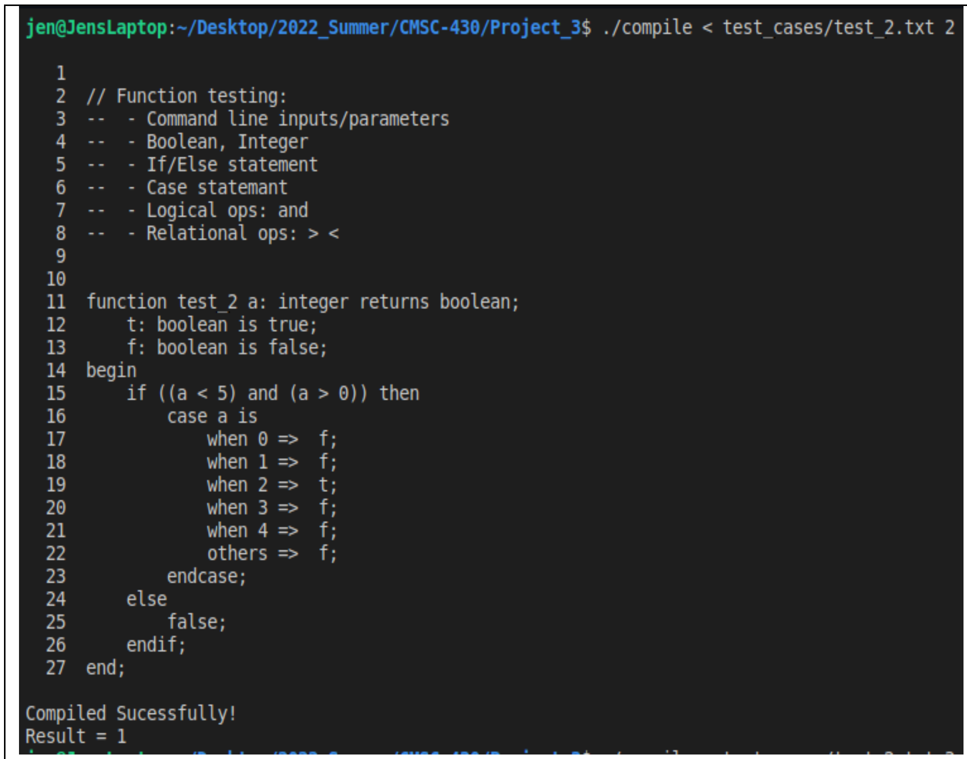
- Test 2



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_2.txt 1

  1
  2  // Function testing:
  3  --  - Command line inputs/parameters
  4  --  - Boolean, Integer
  5  --  - If/Else statement
  6  --  - Case statemant
  7  --  - Logical ops: and
  8  --  - Relational ops: > <
  9
 10
 11  function test_2 a: integer returns boolean;
 12      t: boolean is true;
 13      f: boolean is false;
 14  begin
 15      if ((a < 5) and (a > 0)) then
 16          case a is
 17              when 0 =>  f;
 18              when 1 =>  f;
 19              when 2 =>  t;
 20              when 3 =>  f;
 21              when 4 =>  f;
 22              others =>  f;
 23          endcase;
 24      else
 25          false;
 26      endif;
 27  end;

Compiled Sucessfully!
Result = 0
```

(Passing parameter a=1) This test should enter the if condition, then enter the case condition. Where a meets 'when 1' (Line 18) will return f (f=false).

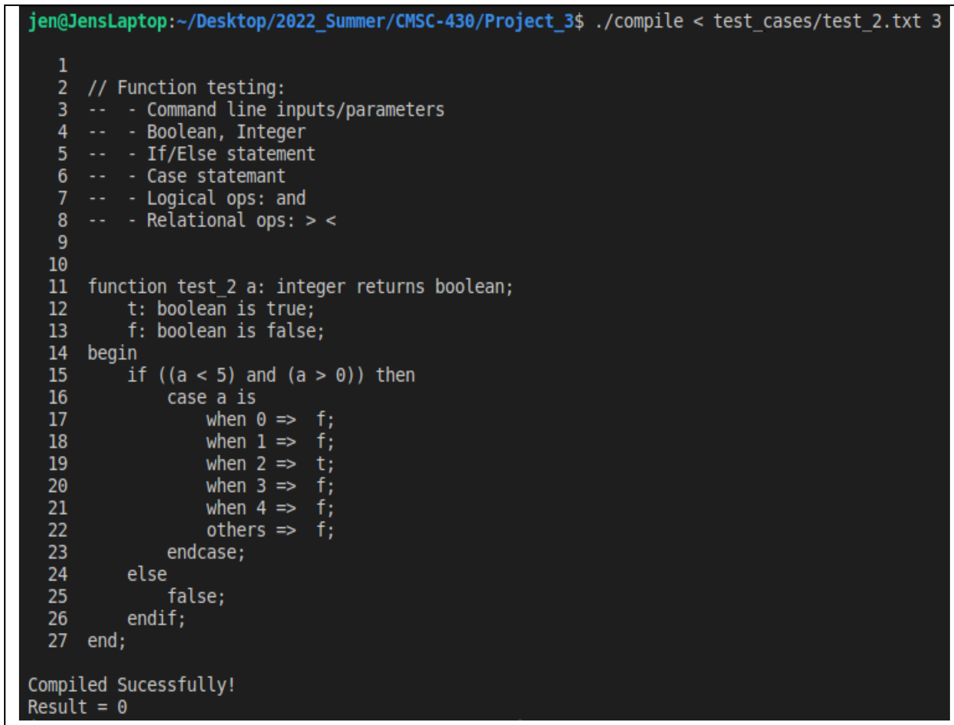
- Test 3



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_2.txt 2

  1
  2  // Function testing:
  3  --  - Command line inputs/parameters
  4  --  - Boolean, Integer
  5  --  - If/Else statement
  6  --  - Case statemant
  7  --  - Logical ops: and
  8  --  - Relational ops: > <
  9
 10
 11  function test_2 a: integer returns boolean;
 12      t: boolean is true;
 13      f: boolean is false;
 14  begin
 15      if ((a < 5) and (a > 0)) then
 16          case a is
 17              when 0 =>  f;
 18              when 1 =>  f;
 19              when 2 =>  t;
 20              when 3 =>  f;
 21              when 4 =>  f;
 22              others =>  f;
 23          endcase;
 24      else
 25          false;
 26      endif;
 27  end;

Compiled Sucessfully!
Result = 1
```

(Passing parameter a=2) This test should enter the if condition, then enter the case condition. Where a meets 'when 2' (Line 19) will return t (t=true).

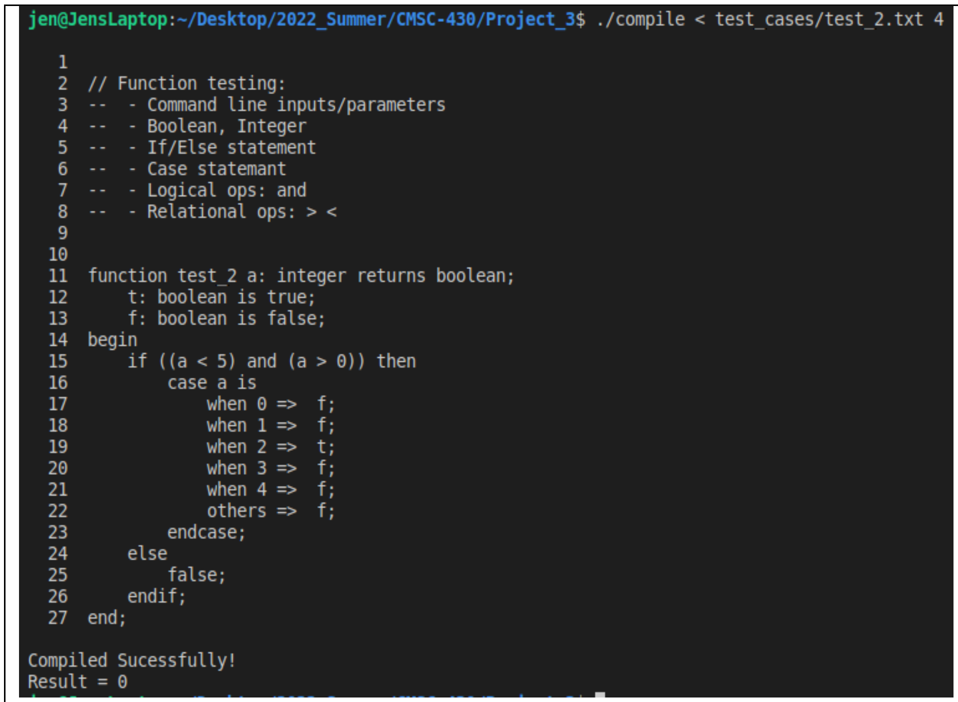
- Test 4



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_2.txt 3

  1
  2  // Function testing:
  3  --  - Command line inputs/parameters
  4  --  - Boolean, Integer
  5  --  - If/Else statement
  6  --  - Case statemant
  7  --  - Logical ops: and
  8  --  - Relational ops: > <
  9
 10
 11  function test_2 a: integer returns boolean;
 12      t: boolean is true;
 13      f: boolean is false;
 14  begin
 15      if ((a < 5) and (a > 0)) then
 16          case a is
 17              when 0 =>  f;
 18              when 1 =>  f;
 19              when 2 =>  t;
 20              when 3 =>  f;
 21              when 4 =>  f;
 22              others =>  f;
 23          endcase;
 24      else
 25          false;
 26      endif;
 27  end;

Compiled Sucessfully!
Result = 0
```

(Passing parameter a=3) This test should enter the if condition, then enter the case condition. Where a meets 'when 3' (Line 20) will return f (f=false).

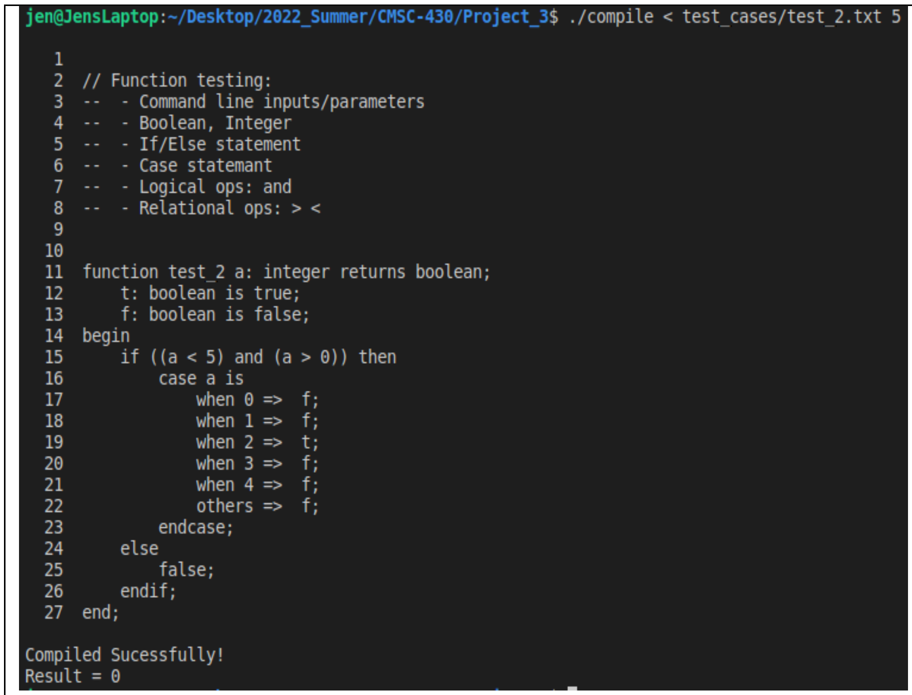
- Test 5



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_2.txt 4

  1
  2  // Function testing:
  3  --  - Command line inputs/parameters
  4  --  - Boolean, Integer
  5  --  - If/Else statement
  6  --  - Case statemant
  7  --  - Logical ops: and
  8  --  - Relational ops: > <
  9
 10
 11  function test_2 a: integer returns boolean;
 12      t: boolean is true;
 13      f: boolean is false;
 14  begin
 15      if ((a < 5) and (a > 0)) then
 16          case a is
 17              when 0 =>  f;
 18              when 1 =>  f;
 19              when 2 =>  t;
 20              when 3 =>  f;
 21              when 4 =>  f;
 22              others =>  f;
 23          endcase;
 24      else
 25          false;
 26      endif;
 27  end;

Compiled Sucessfully!
Result = 0
```

(Passing parameter a=4) This test should enter the if condition, then enter the case condition. Where a meets 'when 4' (Line 21) will return f (f=false).

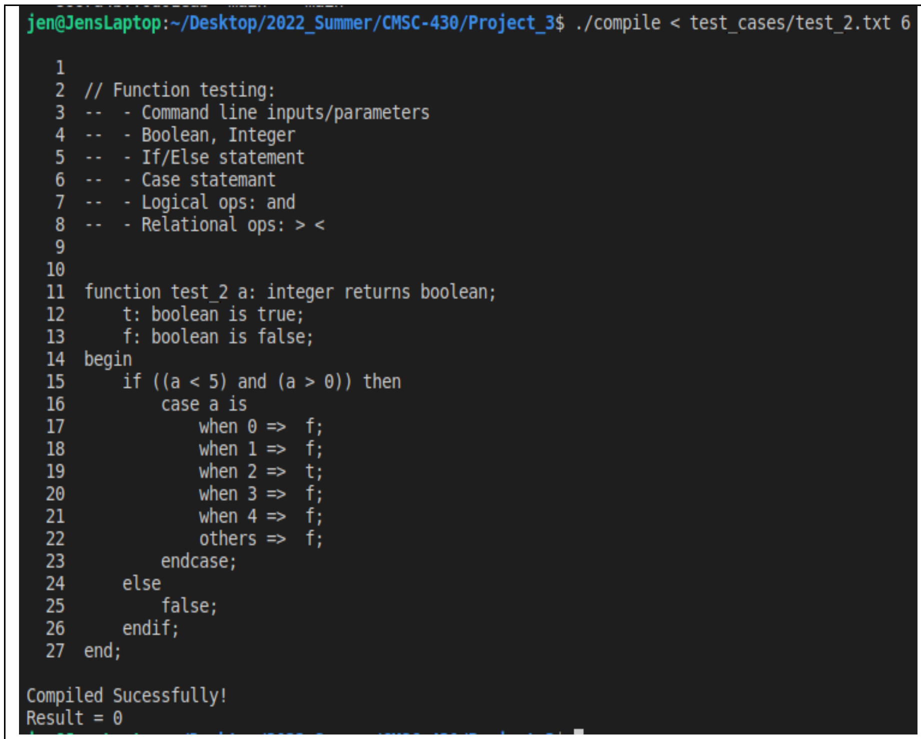
- Test 6



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_2.txt 5

   1
   2  // Function testing:
   3  --   - Command line inputs/parameters
   4  --   - Boolean, Integer
   5  --   - If/Else statement
   6  --   - Case statemant
   7  --   - Logical ops: and
   8  --   - Relational ops: > <
   9
  10
  11  function test_2 a: integer returns boolean;
  12      t: boolean is true;
  13      f: boolean is false;
  14  begin
  15      if ((a < 5) and (a > 0)) then
  16          case a is
  17              when 0 =>  f;
  18              when 1 =>  f;
  19              when 2 =>  t;
  20              when 3 =>  f;
  21              when 4 =>  f;
  22              others =>  f;
  23          endcase;
  24      else
  25          false;
  26      endif;
  27  end;

Compiled Sucessfully!
Result = 0
```

(Passing parameter a=5) This test should enter the if condition, then enter the case condition. Where a meets 'others' (Line 22) will return f (f=false).

- Test 7



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_2.txt 6

   1
   2  // Function testing:
   3  --   - Command line inputs/parameters
   4  --   - Boolean, Integer
   5  --   - If/Else statement
   6  --   - Case statemant
   7  --   - Logical ops: and
   8  --   - Relational ops: > <
   9
  10
  11  function test_2 a: integer returns boolean;
  12      t: boolean is true;
  13      f: boolean is false;
  14  begin
  15      if ((a < 5) and (a > 0)) then
  16          case a is
  17              when 0 =>  f;
  18              when 1 =>  f;
  19              when 2 =>  t;
  20              when 3 =>  f;
  21              when 4 =>  f;
  22              others =>  f;
  23          endcase;
  24      else
  25          false;
  26      endif;
  27  end;

Compiled Sucessfully!
Result = 0
```

(Passing parameter a=6) This test should enter the else condition (Line 24) will return f (f=false).

Test Case 3:

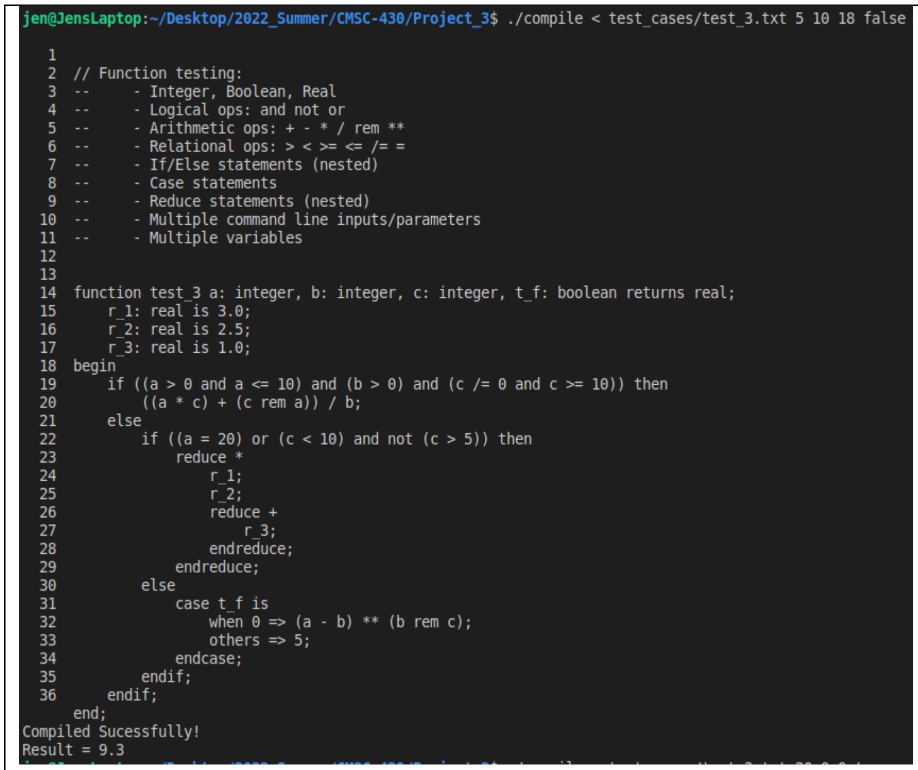
Testing for this test case will cover:

- Integer, Boolean, Real
- Logical ops: and not or
- Arithmetic ops: + - * / rem **
- Relational ops: > < >= <= /= =
- If/Else statements (nested)
- Case statements
- Reduce statements
- Multiple command line inputs/parameters
- Multiple variables

Test Num.	a	b	c	t_f	Result
Test 1	5	10	18	false	9.3
Test 2	20	0	0	true	7.5
Test 3	0	3	7	False	-27
Test 4	5	0	50	True	5

Test Case 3 Screen Shots:

- Test 1



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_3.txt 5 10 18 false

   1
   2  // Function testing:
   3  --      - Integer, Boolean, Real
   4  --      - Logical ops: and not or
   5  --      - Arithmetic ops: + - * / rem **
   6  --      - Relational ops: > < >= <= /= =
   7  --      - If/Else statements (nested)
   8  --      - Case statements
   9  --      - Reduce statements (nested)
  10  --      - Multiple command line inputs/parameters
  11  --      - Multiple variables
  12
  13
  14  function test_3 a: integer, b: integer, c: integer, t_f: boolean returns real;
  15      r_1: real is 3.0;
  16      r_2: real is 2.5;
  17      r_3: real is 1.0;
  18  begin
  19      if ((a > 0 and a <= 10) and (b > 0) and (c /= 0 and c >= 10)) then
  20          ((a * c) + (c rem a)) / b;
  21      else
  22          if ((a = 20) or (c < 10) and not (c > 5)) then
  23              reduce *
  24                  r_1;
  25                  r_2;
  26                  reduce +
  27                      r_3;
  28                  endreduce;
  29              endreduce;
  30          else
  31              case t_f is
  32                  when 0 => (a - b) ** (b rem c);
  33                  others => 5;
  34              endcase;
  35          endif;
  36      endif;
      end;
Compiled Sucessfully!
Result = 9.3
```

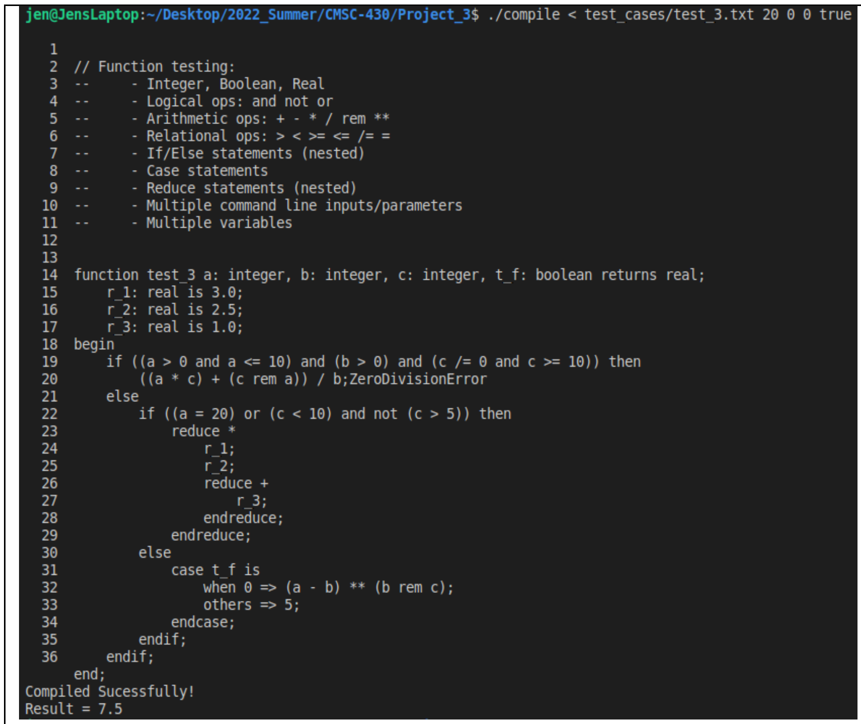
(Passing parameters a=5, b=10, c=18, t_f=false) This test should enter the first if condition (Line 19) and return:

((a * c) + (c rem a)) / b

((5 * 18) + (18 rem 5)) /10

((90) + (3)) / 10 = 9.3

- Test 2



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_3.txt 20 0 0 true

   1
   2  // Function testing:
   3  --      - Integer, Boolean, Real
   4  --      - Logical ops: and not or
   5  --      - Arithmetic ops: + - * / rem **
   6  --      - Relational ops: > < >= <= /= =
   7  --      - If/Else statements (nested)
   8  --      - Case statements
   9  --      - Reduce statements (nested)
  10  --      - Multiple command line inputs/parameters
  11  --      - Multiple variables
  12
  13
  14  function test_3 a: integer, b: integer, c: integer, t_f: boolean returns real;
  15      r_1: real is 3.0;
  16      r_2: real is 2.5;
  17      r_3: real is 1.0;
  18  begin
  19      if ((a > 0 and a <= 10) and (b > 0) and (c /= 0 and c >= 10)) then
  20          ((a * c) + (c rem a)) / b;ZeroDivisionError
  21      else
  22          if ((a = 20) or (c < 10) and not (c > 5)) then
  23              reduce *
  24                  r_1;
  25                  r_2;
  26                  reduce +
  27                      r_3;
  28                  endreduce;
  29              endreduce;
  30          else
  31              case t_f is
  32                  when 0 => (a - b) ** (b rem c);
  33                  others => 5;
  34              endcase;
  35          endif;
  36      endif;
      end;
Compiled Sucessfully!
Result = 7.5
```

(Passing parameters a=20, b=0, c=0, t_f=true) This test should enter the first else condition, then hit the next if condition (Line 22). Where it will return:

reduce *

    3

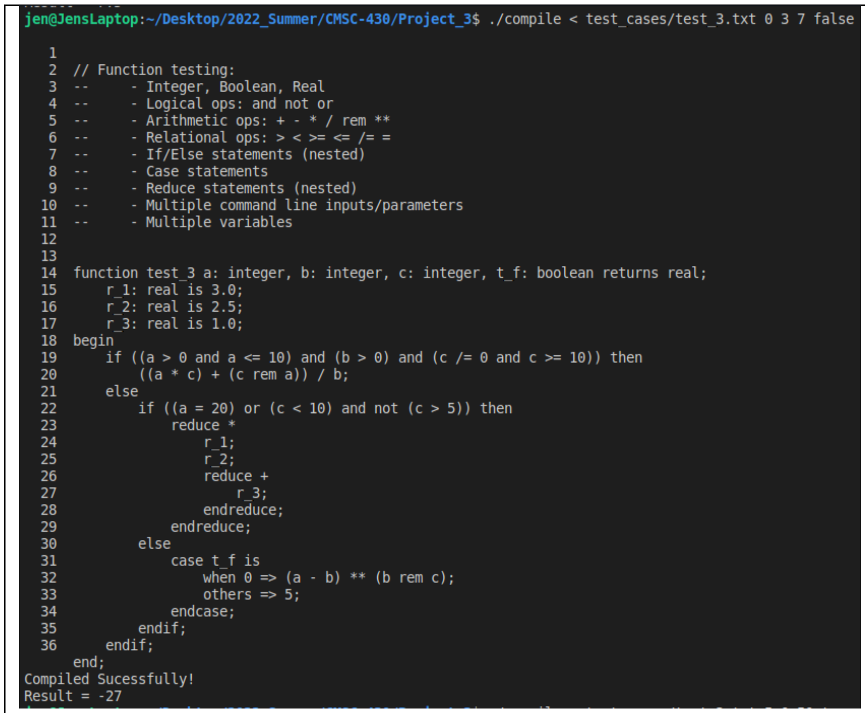
    2.5

    reduce +

        1.0

= 1.0 + (2.5 * 3.0) = 7.5

- Test 3



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_3.txt 0 3 7 false

   1
   2  // Function testing:
   3  --      - Integer, Boolean, Real
   4  --      - Logical ops: and not or
   5  --      - Arithmetic ops: + - * / rem **
   6  --      - Relational ops: > < >= <= /= =
   7  --      - If/Else statements (nested)
   8  --      - Case statements
   9  --      - Reduce statements (nested)
  10  --      - Multiple command line inputs/parameters
  11  --      - Multiple variables
  12
  13
  14  function test_3 a: integer, b: integer, c: integer, t_f: boolean returns real;
  15      r_1: real is 3.0;
  16      r_2: real is 2.5;
  17      r_3: real is 1.0;
  18  begin
  19      if ((a > 0 and a <= 10) and (b > 0) and (c /= 0 and c >= 10)) then
  20          ((a * c) + (c rem a)) / b;
  21      else
  22          if ((a = 20) or (c < 10) and not (c > 5)) then
  23              reduce *
  24                  r_1;
  25                  r_2;
  26                  reduce +
  27                      r_3;
  28                  endreduce;
  29              endreduce;
  30          else
  31              case t_f is
  32                  when 0 => (a - b) ** (b rem c);
  33                  others => 5;
  34              endcase;
  35          endif;
  36      endif;
      end;
Compiled Sucessfully!
Result = -27
```

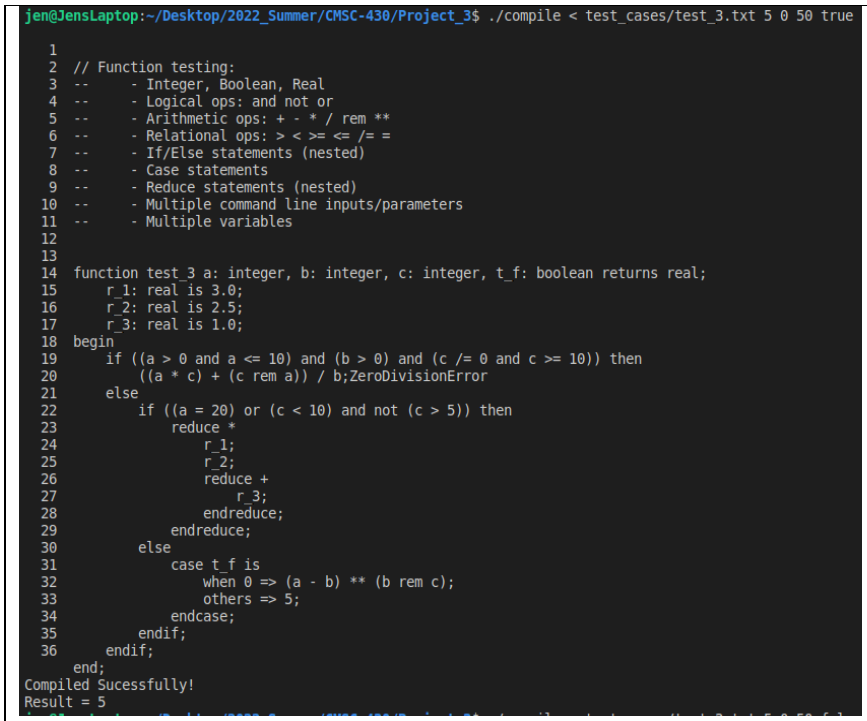
(Passing parameters a=0, b=3, c=7, t_f=false) This test should enter the first else condition, then into the next else condition. Where it will enter the case statement. While t_f=false, it will hit ‘when 0’ (Line 32) and return:

(a - b) ** (b rem c)

(0 - 3) + (3 rem 7)

(-3) ** (3) = -27

- Test 4



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_3.txt 5 0 50 true

   1
   2  // Function testing:
   3  --      - Integer, Boolean, Real
   4  --      - Logical ops: and not or
   5  --      - Arithmetic ops: + - * / rem **
   6  --      - Relational ops: > < >= <= /= =
   7  --      - If/Else statements (nested)
   8  --      - Case statements
   9  --      - Reduce statements (nested)
  10  --      - Multiple command line inputs/parameters
  11  --      - Multiple variables
  12
  13
  14  function test_3 a: integer, b: integer, c: integer, t_f: boolean returns real;
  15      r_1: real is 3.0;
  16      r_2: real is 2.5;
  17      r_3: real is 1.0;
  18  begin
  19      if ((a > 0 and a <= 10) and (b > 0) and (c /= 0 and c >= 10)) then
  20          ((a * c) + (c rem a)) / b;ZeroDivisionError
  21      else
  22          if ((a = 20) or (c < 10) and not (c > 5)) then
  23              reduce *
  24                  r_1;
  25                  r_2;
  26                  reduce +
  27                      r_3;
  28                  endreduce;
  29              endreduce;
  30          else
  31              case t_f is
  32                  when 0 => (a - b) ** (b rem c);
  33                  others => 5;
  34              endcase;
  35          endif;
  36      endif;
      end;
Compiled Sucessfully!
Result = 5
```

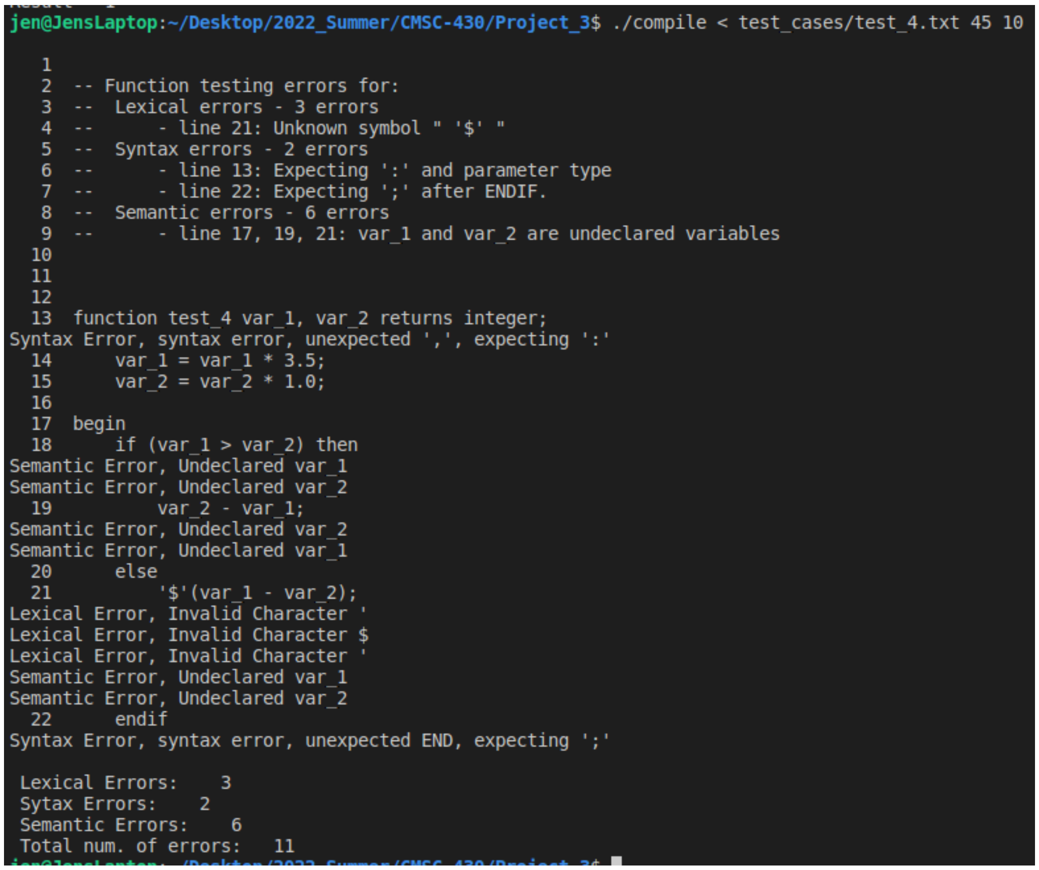
(Passing parameters a=5, b=0, c=50, t_f=true) This test should enter the first else condition, then into the next else condition. Where it will enter the case statement. While t_f=true, it will hit 'others' (Line 33) and return:

5

Test Case 4:

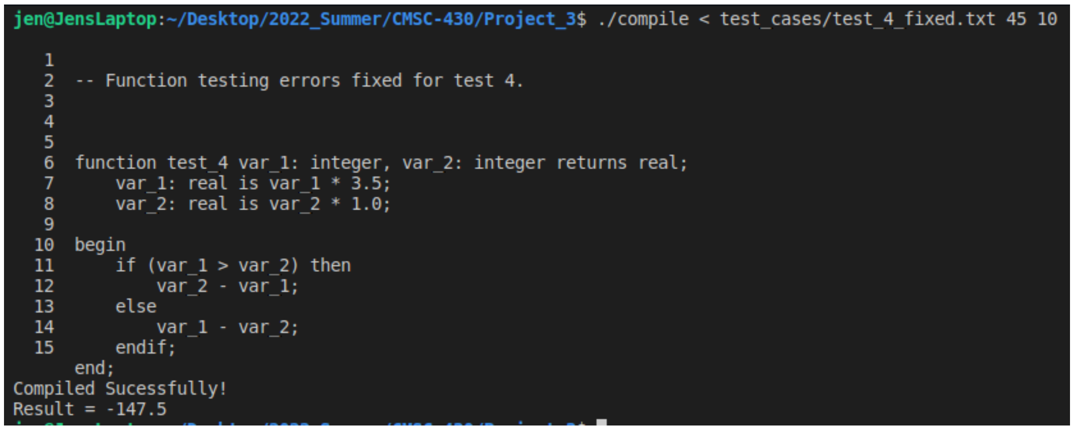
Testing for this test case will test errors for:

- Lexical errors - 3 errors
  - line 21: Unknown symbol " '$' "
- Syntax errors - 2 errors
  - line 13: Expecting ':' and parameter type
  - line 22: Expecting ';' after ENDIF.
- Semantic errors - 6 errors
  - line 17, 19, 21: var_1 and var_2 are undeclared variables



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_4.txt 45 10

    1
    2  -- Function testing errors for:
    3  --  Lexical errors - 3 errors
    4  --       - line 21: Unknown symbol " '$' "
    5  --  Syntax errors - 2 errors
    6  --       - line 13: Expecting ':' and parameter type
    7  --       - line 22: Expecting ';' after ENDIF.
    8  --  Semantic errors - 6 errors
    9  --       - line 17, 19, 21: var_1 and var_2 are undeclared variables
   10
   11
   12
   13  function test_4 var_1, var_2 returns integer;
Syntax Error, syntax error, unexpected ',', expecting ':'
   14      var_1 = var_1 * 3.5;
   15      var_2 = var_2 * 1.0;
   16
   17  begin
   18      if (var_1 > var_2) then
Semantic Error, Undeclared var_1
Semantic Error, Undeclared var_2
   19          var_2 - var_1;
Semantic Error, Undeclared var_2
Semantic Error, Undeclared var_1
   20      else
   21          '$'(var_1 - var_2);
Lexical Error, Invalid Character '
Lexical Error, Invalid Character $
Lexical Error, Invalid Character '
Semantic Error, Undeclared var_1
Semantic Error, Undeclared var_2
   22      endif
Syntax Error, syntax error, unexpected END, expecting ';'

 Lexical Errors:    3
 Sytax Errors:    2
 Semantic Errors:    6
 Total num. of errors:    11
```

Test Case 4 (fixed errors)



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < test_cases/test_4_fixed.txt 45 10


-- Function testing errors fixed for test 4.



function test_4 var_1: integer, var_2: integer returns real;
    var_1: real is var_1 * 3.5;
    var_2: real is var_2 * 1.0;

begin
    if (var_1 > var_2) then
        var_2 - var_1;
    else
        var_1 - var_2;
    endif;
end;
Compiled Sucessfully!
Result = -147.5
```

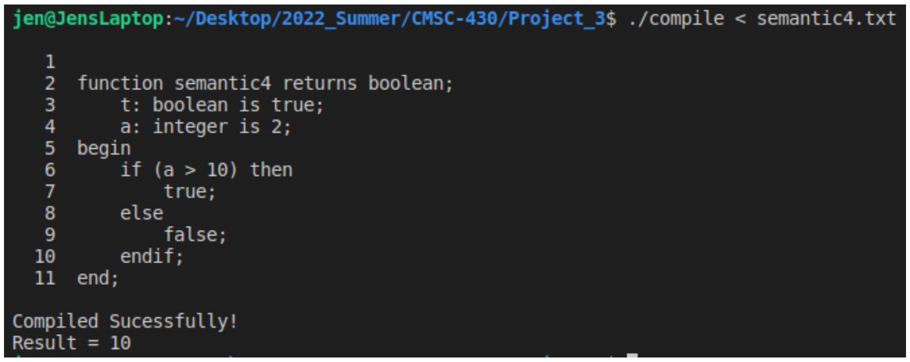
Lessons Learned

Project three was not only the most difficult project for the course so far, it was also the most time consuming, and required more thought. While that may sound bad, I really enjoy projects that challenge me and require some critical thinking, and problem solving.

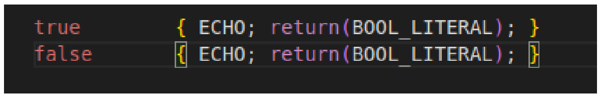
Starting project three seemed pretty easy. The videos and course content really help explain how the parser.y, symbols.h, and the value.cc files communicate. At first when looking at the syntax for the semantic actions in the parser.y file, I got a little confused on the structure and had to read more about what the rules were doing and how to construct them on my own. Once I figured that out, I went ahead and got the easier rules out of the way. As I figured writing the if statement and the case statement would probably take more time.

After I got the relational, logical, and mathematical operations finished I did a couple of tests to make sure they were working. Unfortunately, I don't have any screen shots at this stage of the project. However, running the semantic4.txt that was included in our skeleton code worked great! The math operators were calculating the correct result and it was reading the program fine. Although, the only operators the program tested was "(b + 2) * (2 + 4)" and that returned a Boolean. So, no parameters, variables, and only a couple basic math operations. Needless to say, when started testing more complex programs that required command line arguments, Boolean variables, and math operators (specifically "rem") things started breaking.

Before I got started on my case and if statements, I wanted to get all my operators working properly. Figuring out how to read a Boolean variable was probably the easiest thing to tackle first. I figured it had something to do with how it was being labeled in the scanner.l file. I found out I was not setting a value for either of the BOOL_LITERALs in the file. In the screen shot below, you can see I was getting some pretty weird results. As this program should have returned either a '1' or a '0'.

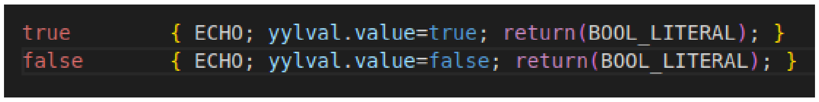


```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < semantic4.txt


function semantic4 returns boolean;
    t: boolean is true;
    a: integer is 2;
begin
    if (a > 10) then
        true;
    else
        false;
    endif;
end;

Compiled Sucessfully!
Result = 10
```

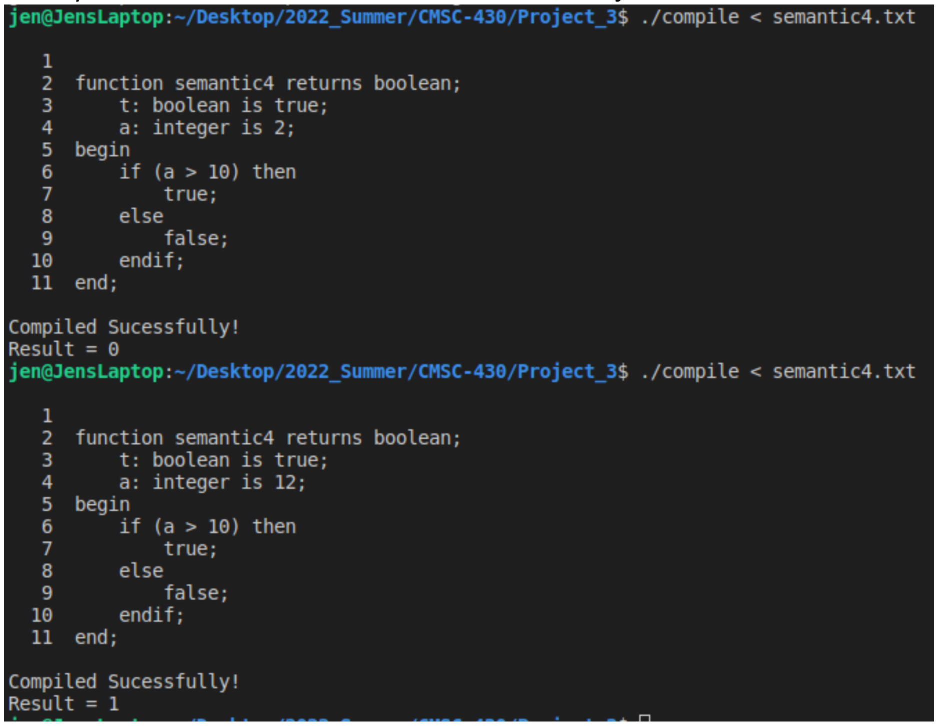


```
true        { ECHO; return(BOOL_LITERAL); }
false       { ECHO; return(BOOL_LITERAL); }
```

After adding in a value for both BOOL_LITERALs, I got a more correct looking result.

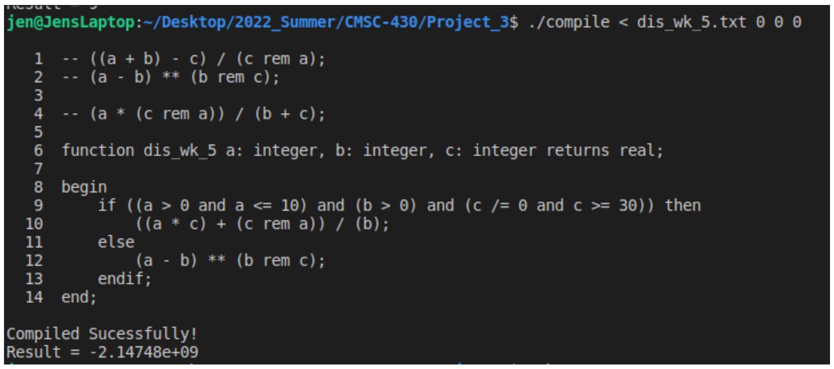


```
true        { ECHO; yylval.value=true; return(BOOL_LITERAL); }
false       { ECHO; yylval.value=false; return(BOOL_LITERAL); }
```

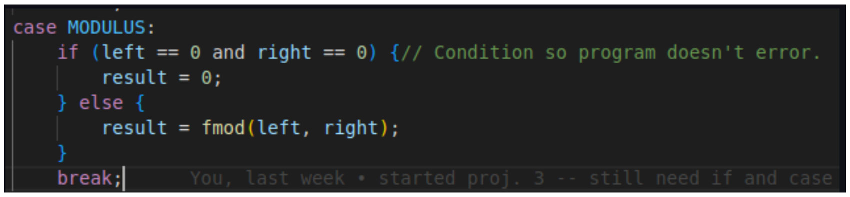


```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < semantic4.txt


function semantic4 returns boolean;
    t: boolean is true;
    a: integer is 2;
begin
    if (a > 10) then
        true;
    else
        false;
    endif;
end;

Compiled Sucessfully!
Result = 0
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < semantic4.txt


function semantic4 returns boolean;
    t: boolean is true;
    a: integer is 12;
begin
    if (a > 10) then
        true;
    else
        false;
    endif;
end;

Compiled Sucessfully!
Result = 1
```

The next step was to try and figure out why the “rem” operator was giving me crazy results. In the screen shot, the result should be returning zero as I am passing 0, 0, 0. Instead the result kept returning -214748e+09. I really do not know why this was happening, but I figured id add a condition to the operator case to return zero if both left and right were equal to zero. That seemed to have fixed the issue.



```
jen@JensLaptop:~/Desktop/2022_Summer/CMSC-430/Project_3$ ./compile < dis_wk_5.txt 0 0 0

-- ((a + b) - c) / (c rem a);
-- (a - b) ** (b rem c);

-- (a * (c rem a)) / (b + c);

function dis_wk_5 a: integer, b: integer, c: integer returns real;

begin
    if ((a > 0 and a <= 10) and (b > 0) and (c /= 0 and c >= 30)) then
        ((a * c) + (c rem a)) / (b);
    else
        (a - b) ** (b rem c);
    endif;
end;

Compiled Sucessfully!
Result = -2.14748e+09
```

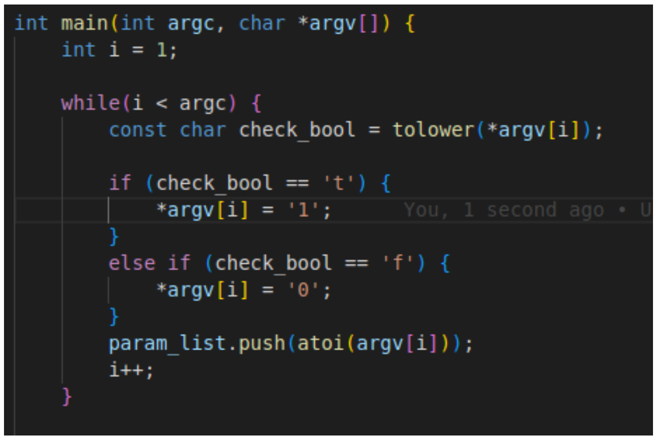


```
case MODULUS:
    if (left == 0 and right == 0) {// Condition so program doesn't error.
        result = 0;
    } else {
        result = fmod(left, right);
    }
    break;      You, last week • started proj. 3 -- still need if and case
```

Once every operator seemed to be working and returning correct results, I got started on the harder stuff. The first thing I started on was the if/else statements. This was surprisingly easier than I thought it would be. Because the expression of an if condition returns either true or false, I could just say if the if condition is equal to ‘1’, then return the if condition. Otherwise return the else condition. That was about it for the if statement.

The case statement on the other hand took me a few days to figure out. I kept getting an “undeclared case”, or a syntax error. While I knew the problem, it was the solution to that problem I was having trouble wrapping my head around. Inevitably, I had to go back and review our course content, along with some outside research. Basically, I knew I needed a couple of functions in the value.cc files that would return the matching case condition (the “when”/”others”), it was the logic I was having a problem with. I finally figured out I could set the case condition (the variable in question), before calling the evaluate_caseStat(). Then, I created global variables in the value.cc that would house both, the case condition and the matching case statement (if there was one, or returning others statement). At this point, I thought I had finished my program and was really happy to get everything finished so early. However, while testing out other student’s code I found an error in the program (a logic error), that was not returning an error, the program would still compile but I had not noticed it until I ran another student’s code.

My biggest problem was being able to pass a Boolean as a parameter in the command line. After a bit of looking through my code and running some tests, I found out it was the queue I was using to push my command line argument into. My queue was a double queue, and when it saw ‘true’ or ‘false’ the program could not configure these as doubles, so it would just return ‘0’. I added a few print statements to see where the issue was. Was it coming from the command line itself? Or was it because of the double queue I was using? The print statements were kind of a life saver here. They allowed me to quickly narrow down the issue and figure out how to fix it. Although, fixing this issue took a lot longer than I had anticipated. At first, I figured I would just add a condition in the main that would look for “true” or “false” and return either a 1 or 0. Which in my mind should have worked, but it didn’t. For the next few days, I scoured the internet for some kind of solution. Everything I found was leading me to more and more difficult, or complicated solutions. I had almost given up on fixing this issue, until I came across a forum on stackoverflow, where someone had a similar issue. The answer was in the quotations. I think the biggest lesson I learned from this project is how C++ uses quotation. Going back to my original solution to this problem (using a condition to look for “true” or “false”), I had used double quotations in stead of using single quotations when comparing chars or *chars. I also learned the difference between chars and *chars. After, all of that I could finally say my program was complete, and I was happy I did not give up trying to find a solution. This is a screen shot of the final result to read a bool from the command line:


```
int main(int argc, char *argv[]) {
    int i = 1;

    while(i < argc) {
        const char check_bool = tolower(*argv[i]);

        if (check_bool == 't') {
            *argv[i] = '1';       You, 1 second ago • U
        }
        else if (check_bool == 'f') {
            *argv[i] = '0';
        }
        param_list.push(atoi(argv[i]));
        i++;
    }
```