Project 4

Terrence Jackson

CMSC 430: Compiler Theory and Design

Professor McDonald

October 8, 2024

Approach

To begin this project, I ported over my changes from Project 2. During this process, I also implemented the real literal and hexadecimal literals, following the patterns already established for integer and character literals.

For type checking in list elements, I modeled my implementation after the checkCases() function, making minor modifications. Similarly, for list initialization, I reused the existing checkAssignment() function instead of creating a new one in types.cc, as it already provided the necessary functionality. To confirm this, I ran a test case to ensure valid lists compiled successfully.

Next, I wrote a generic checkType() function to validate that list subscripts are integers, anticipating its broader utility for future validations. Moving on to relational operators, I initially created a function similar to checkAssignment(). However, I quickly realized that relations are part of the arithmetic operator cascade and needed a type for negation validation. I revised my function to first validate character types before calling checkArithmetic(), which handles other types.

The implementation of the if statement's type-checking followed the model of case statements and was relatively straightforward. For fold statements, I realized the need for two generic checkNumeric() functions—one for single values and one for pairs. Since the pair validation was already implemented in checkArithmetic(), I repurposed it to be more flexible by making the error message a parameter and using it in various productions.

When I began validating narrowing, I found a bug in my checkIf() function that threw errors when there were no elsif branches. I corrected this by allowing the elsif type to be null. The final challenge was handling duplicate variables, where I initially attempted an inline solution. I ultimately created a dedicated function for this purpose within the parser.y file, due to its close relationship with the Symbol class.

Test Plan

Semantic	Input	Expected Output	Pass?

semantic1	// Variable Initialization Mismatch
	
	function main returns integer;
	    value: integer is 'A';
	begin
	    1;
	end;
		Semantic Error, Type Mismatch on Variable Initialization
		
		Semantic Errors 1
			Y

semantic2	// When Types Mismatch
	
	function main returns integer;
	begin
	    when 2 < 1, 1 : 'a';
	end;
		Semantic Error, When Types Mismatch
		
		Semantic Errors 1
			Y

semantic3	// Non Integer Switch Expression
	
	function main returns integer;
	    b: character is 'A';
	begin
	    switch b is
	    case 1 => 2;
	    case 2 => 4;
	    others => 6;
	    endswitch;
	End;
		Semantic Error, Switch Expression Not Integer
		
		Semantic Errors 1
			Y

semantic4	// Case Types Mismatch
	
	function main returns integer;
	  b: character is 'b';
	begin
	  switch 1 is
	  case 1 => 2;
	  case 2 => b;
	  others => 6;
	  endswitch;
	end;
		Semantic Error, Case Types Mismatch
		
		Semantic Errors 1
			Y

semantic5	// Using Character Variable with Arithmetic Operator
	
	function main returns integer;
	  b: character is 'b';
	begin
	  b + 10;
	end;
		Semantic Error, Arithmetic Operator Requires Numeric Types
		
		Semantic Errors 1
			Y

semantic6	// Undeclared Scalar Variable
	
	function main returns integer;
	begin
	  2 * b + 3;
	end;
		Semantic Error, Undeclared Scalar b
		
		Semantic Errors 1
			Y

semantic7	// Undeclared List Variable
	
	function main returns integer;
	begin
	  primes(1) + 1;
	end;
		Semantic Error, Undeclared List primes
		
		Semantic Errors 1
			Y

semantic8	// List with Elements of Different Types
	
	function main returns integer;
	  aList: list of integer is (1, 2, 3.5);
	begin
	  aList(1);
	end;
		Semantic Error, List Element Types Do Not Match
		
		Semantic Errors 1
			Y

semantic9	// List Type Does Not Match Element Types
	
	function main returns character;
	        aList: list of character is (1, 2, 3);
	begin
	        aList(1);
	end;
		Semantic Error, List Type Does Not Match Element Types
		
		Semantic Errors 1
			Y

semantic10	// List Subscript is not Integer
	
	function main returns integer;
	        aList: list of integer is (1, 2, 3);
	begin
	        aList(1.5);
	end;
		Semantic Error, List Subscript Must Be Integer
		
		Semantic Errors 1
			Y

semantic11	-- Mixing Numeric and Character Types with Relational Operator
	
	function main returns integer;
	begin
	        if 'b' < 'c' then
	        1;
	        elsif 'b' < 1 then
	        2;
	        else
	        3;
	        endif;
	end;
		Semantic Error, Character Literals Cannot be Compared to Numeric Expressions
		
		Semantic Errors 1
			Y

semantic12	// Using Character Literal with Exponentiation Operator
	// and Negation Operator
	
	function main returns integer;
	        c: character is ~'c';
	begin
	        5 ^ 'P';
	end;
		Semantic Error, Arithmetic Operator Requires Numeric Types
		
		Semantic Error, Arithmetic Operator Requires Numeric Types
		
		Semantic Errors 2
			Y

semantic13	// Mixing Real Literals with the Remainder Operator
	
	function main returns integer;
	begin
	        4 % 4.8;
	end;
		Semantic Error, Remainder Operator Requires Integer Operands
		
		Semantic Errors 1
			Y

semantic14	-- If Elsif Else Mismatch
	
	function main returns integer;
	begin
	        if 9 < 10 then
	        1;
	        elsif 8 = 1 then
	        2;
	        else
	        3.7;
	        endif;
	end;
		Semantic Error, If-Elsif-Else Type Mismatch
		
		Semantic Errors 1
			Y

semantic15	// Folding a nonnumeric List
	
	function main returns integer;
	begin
	        fold left + ('a', 'b', 'c') endfold;
	end;
		Semantic Error, Fold Requires A Numeric List
		
		Semantic Errors 1
			Y

semantic16	-- Narrowing Variable Initialization
	
	function main returns integer;
	        b: integer is 5 * 2.5;
	begin
	        b + 1;
	end;
		Semantic Error, Illegal Narrowing Variable Initialization
		
		Semantic Errors 1
			Y

semantic17	-- Narrowing Function Return
	
	function main returns integer;
	        b: integer is 6 * 2;
	begin
	        if 8 < 0 then
	        b + 3.0;
	        else
	        b * 4.6;
	        endif;
	end;
		Semantic Error, Illegal Narrowing Function Return
		
		Semantic Errors 1
			Y

semantic18	-- Duplicate Scalar and List Variables
	
	function main returns integer;
	        scalar: integer is 4 * 2;
	        scalar: character is 'b';
	        a_list: list of integer is (4, 2);
	        a_list: list of real is (2.3, 4.4);
	begin
	        1;
	end;
		Semantic Error, Duplicate Scalar scalar
		
		Semantic Error, Duplicate List a_list
		
		Semantic Errors 2
			Y

semantic19	// Multiple Semantic Errors
	
	function main returns integer;
	        value: integer is 4.5;
	        numbers: list of real is (1, 2, 3);
	        one: integer is '1';
	begin
	        if value > 0 then
	        fold left + ('a', 'b') endfold;
	        elsif name = 'N' then
	        fold right * (1, 2.5) endfold;
	        else
	        when value < 10, 1 : 1.5;
	        endif;
	end;
		Semantic Error, Illegal Narrowing Variable Initialization
		
		Semantic Error, List Type Does Not Match Element Types
		
		Semantic Error, Type Mismatch on Variable Initialization
		
		Semantic Error, Fold Requires A Numeric List
		Semantic Error, Undeclared Scalar name
		
		Semantic Error, List Element Types Do Not Match
		
		Semantic Error, When
			Y

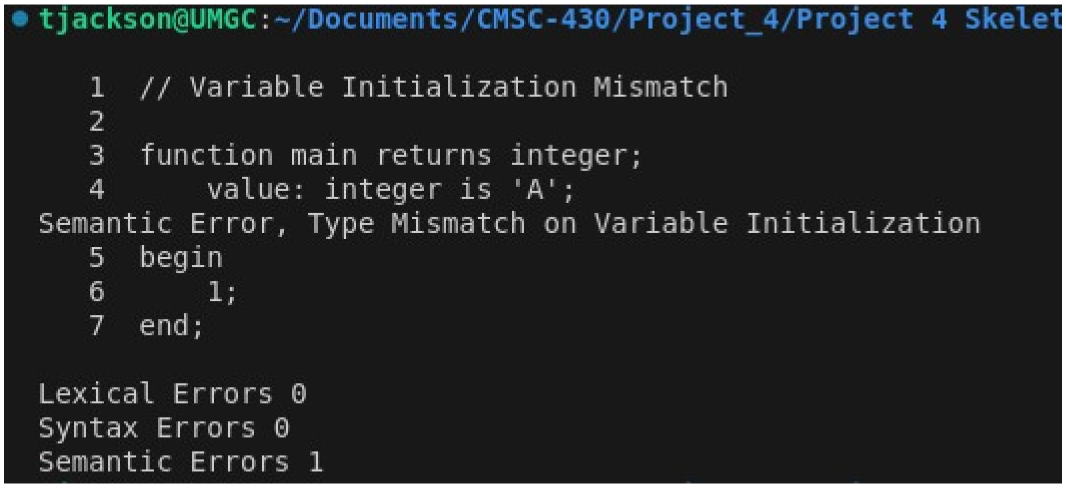
		Types Mismatch Semantic Errors 7	

valid1	-- Program with a Real Variable
	
	function main returns real;
	    a: real is 4.5;
	begin
	    a;
	end;
		Compiled Successfully	Y

valid2	-- Program with a Hexadecimal Literals
	
	function main returns integer;
	    a: integer is #A;
	begin
	    a + #a;
	end;
		Compiled Successfully	Y

valid3	-- Program with a Real Variable
	
	function main returns real;
	    a: real is 4 + 4.5;
	begin
	    a;
	end;
		Compiled Successfully	Y

Semantic 1:

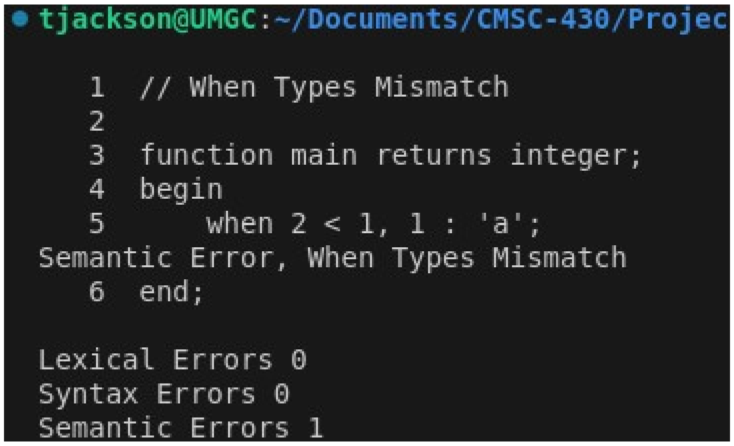


```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project 4 Skelet

   1  // Variable Initialization Mismatch
   2
   3  function main returns integer;
   4      value: integer is 'A';
Semantic Error, Type Mismatch on Variable Initialization
   5  begin
   6      1;
   7  end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 2:



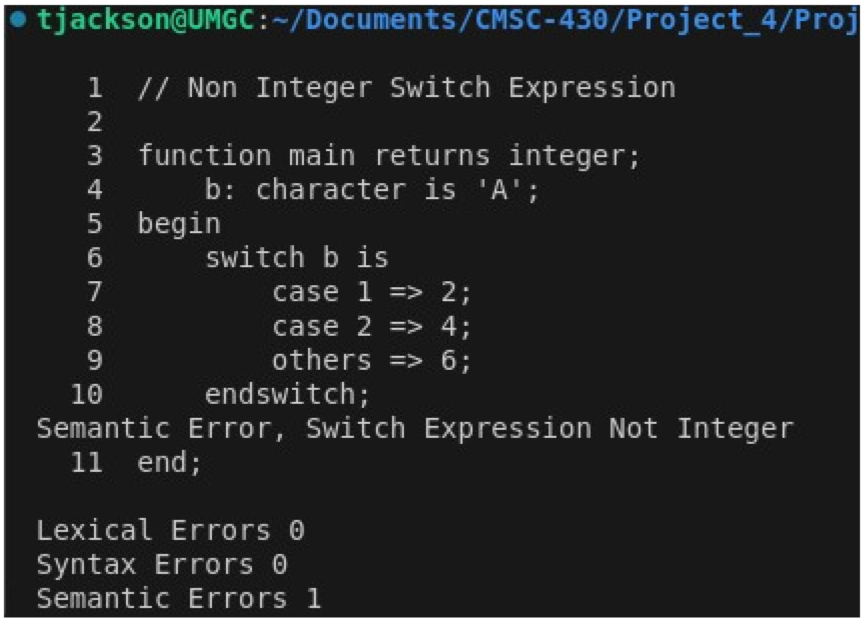
```
● tjackson@UMGC:~/Documents/CMSC-430/Projec

   // When Types Mismatch

   function main returns integer;
   begin
       when 2 < 1, 1 : 'a';
Semantic Error, When Types Mismatch
   end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 3:



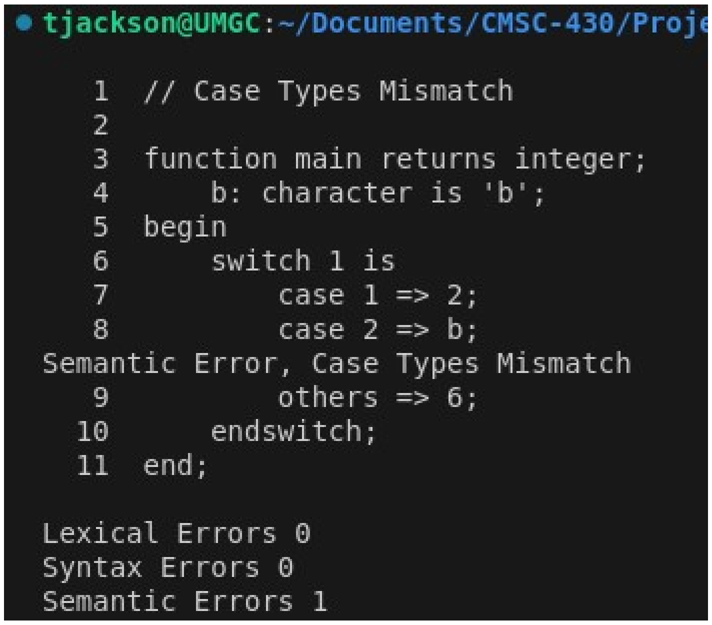
```
● tjackson@UMGC:~/Documents/CMSC-430/Project_4/Proj

   // Non Integer Switch Expression

   function main returns integer;
       b: character is 'A';
   begin
       switch b is
           case 1 => 2;
           case 2 => 4;
           others => 6;
       endswitch;
Semantic Error, Switch Expression Not Integer
   end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 4:



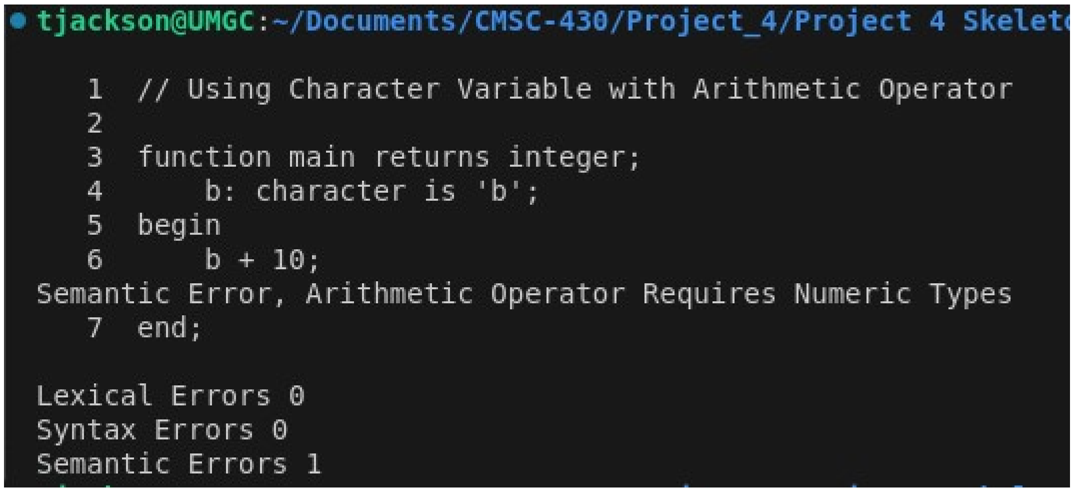
```
● tjackson@UMGC:~/Documents/CMSC-430/Proje

   // Case Types Mismatch

   function main returns integer;
       b: character is 'b';
   begin
       switch 1 is
           case 1 => 2;
           case 2 => b;
Semantic Error, Case Types Mismatch
           others => 6;
       endswitch;
   end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 5:



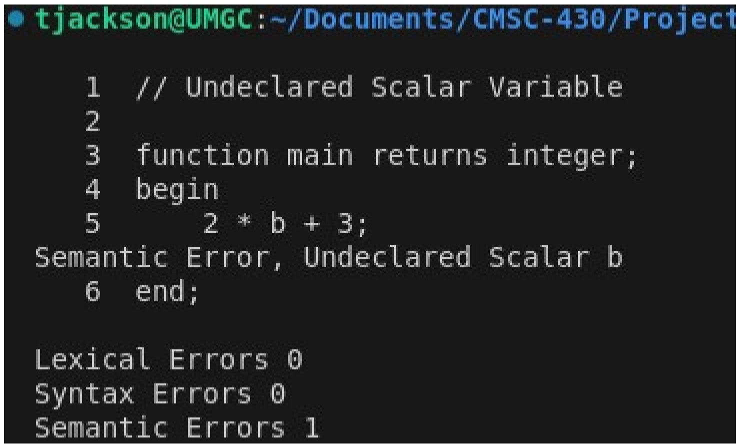
```
● tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project 4 Skelet

   // Using Character Variable with Arithmetic Operator

   function main returns integer;
       b: character is 'b';
   begin
       b + 10;
Semantic Error, Arithmetic Operator Requires Numeric Types
   end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 6:



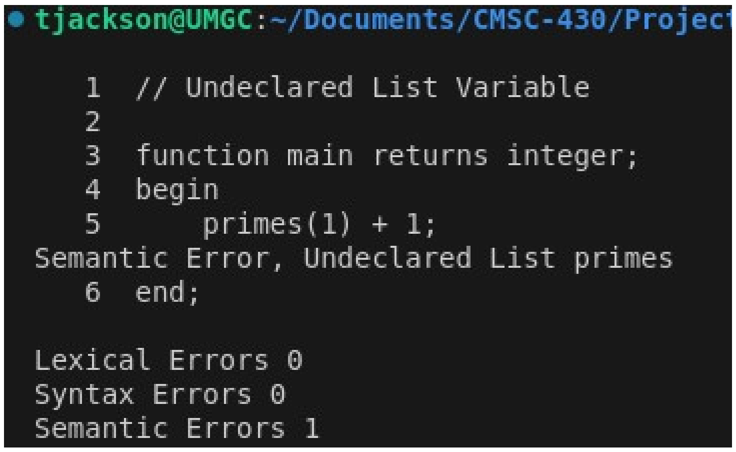
```
● tjackson@UMGC:~/Documents/CMSC-430/Project

  // Undeclared Scalar Variable

  function main returns integer;
  begin
      2 * b + 3;
Semantic Error, Undeclared Scalar b
  end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 7:



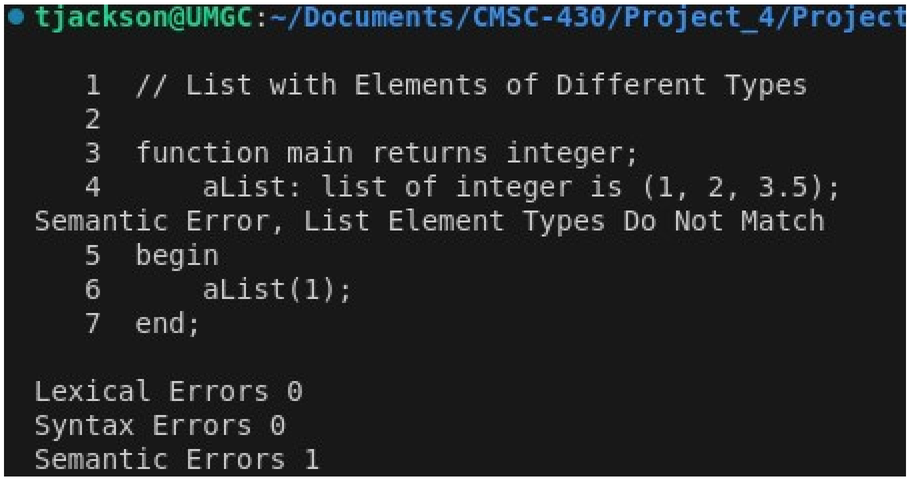
```
● tjackson@UMGC:~/Documents/CMSC-430/Project

  // Undeclared List Variable

  function main returns integer;
  begin
      primes(1) + 1;
Semantic Error, Undeclared List primes
  end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 8:



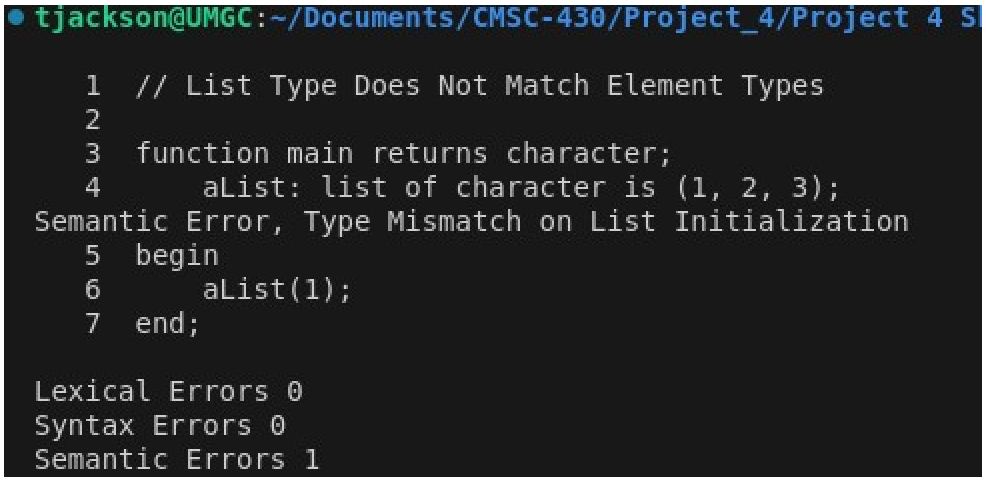
```
● tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project

  // List with Elements of Different Types

  function main returns integer;
      aList: list of integer is (1, 2, 3.5);
Semantic Error, List Element Types Do Not Match
  begin
      aList(1);
  end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 9:



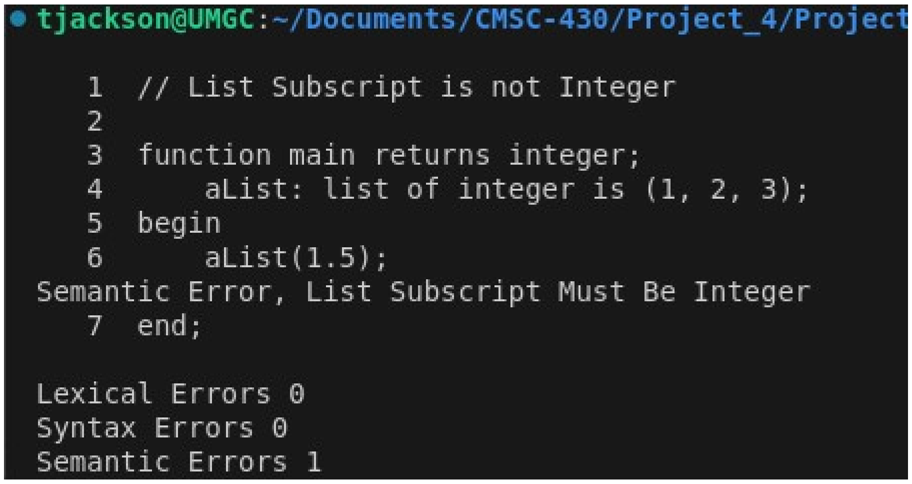
```
● tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project 4 Sl

  // List Type Does Not Match Element Types

  function main returns character;
      aList: list of character is (1, 2, 3);
Semantic Error, Type Mismatch on List Initialization
  begin
      aList(1);
  end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 10:



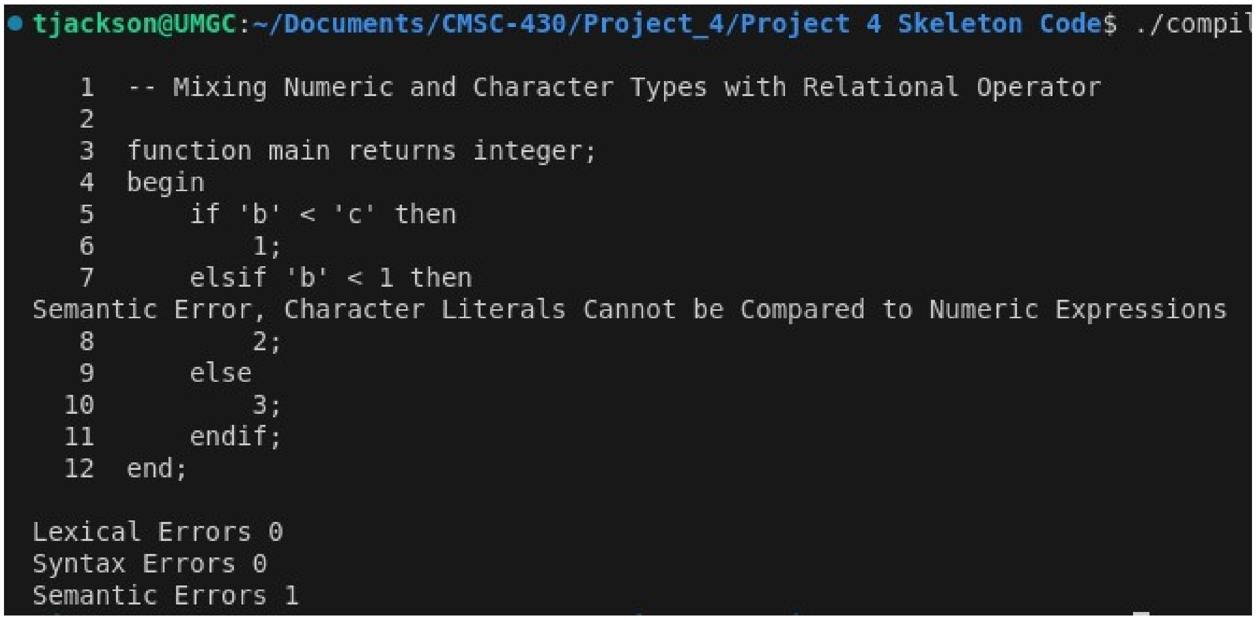
```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project

  // List Subscript is not Integer

  function main returns integer;
      aList: list of integer is (1, 2, 3);
  begin
      aList(1.5);
Semantic Error, List Subscript Must Be Integer
  end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 11:



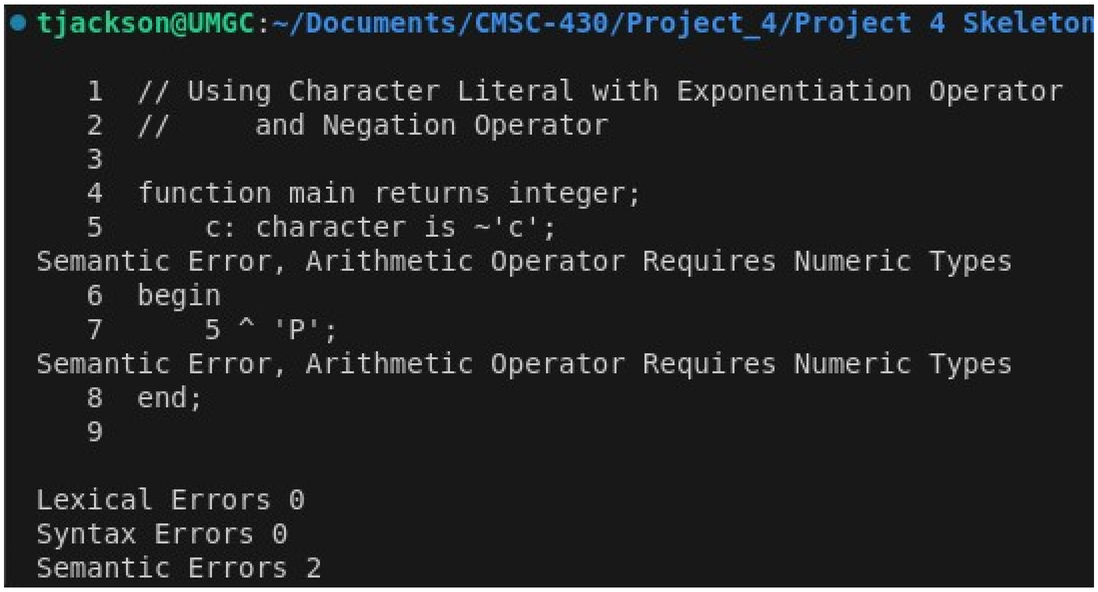
```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project 4 Skeleton Code$ ./compil

  -- Mixing Numeric and Character Types with Relational Operator

  function main returns integer;
  begin
      if 'b' < 'c' then
          1;
      elsif 'b' < 1 then
Semantic Error, Character Literals Cannot be Compared to Numeric Expressions
          2;
      else
          3;
      endif;
  end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 12:



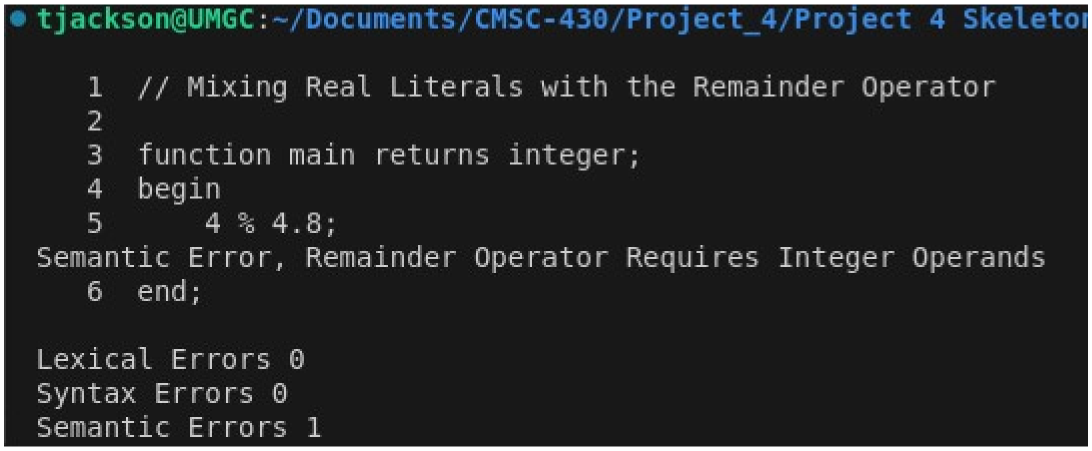
```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project 4 Skeleton

   // Using Character Literal with Exponentiation Operator
   //     and Negation Operator

   function main returns integer;
       c: character is ~'c';
Semantic Error, Arithmetic Operator Requires Numeric Types
   begin
       5 ^ 'P';
Semantic Error, Arithmetic Operator Requires Numeric Types
   end;


Lexical Errors 0
Syntax Errors 0
Semantic Errors 2
```

Semantic 13:



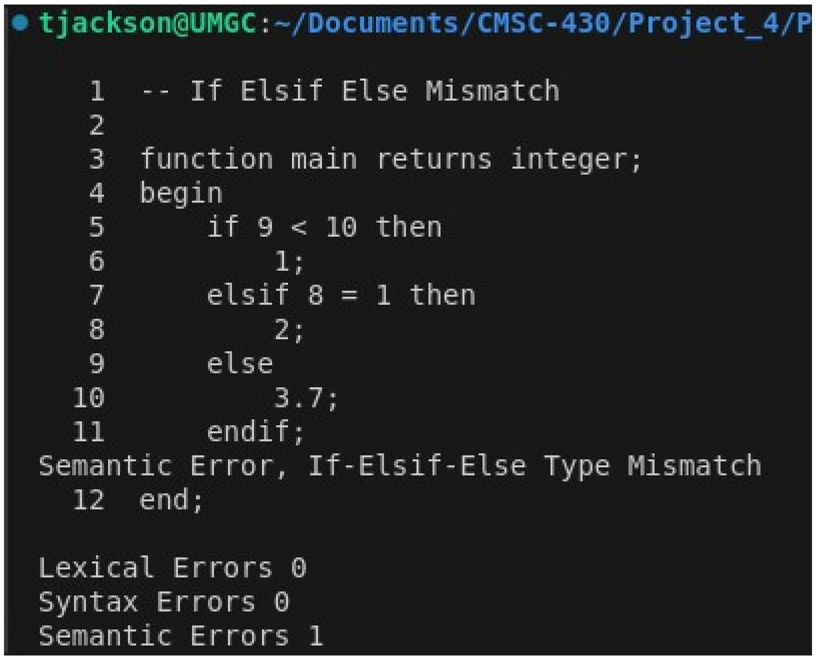
```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project 4 Skeleton

   // Mixing Real Literals with the Remainder Operator

   function main returns integer;
   begin
       4 % 4.8;
Semantic Error, Remainder Operator Requires Integer Operands
   end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 14:



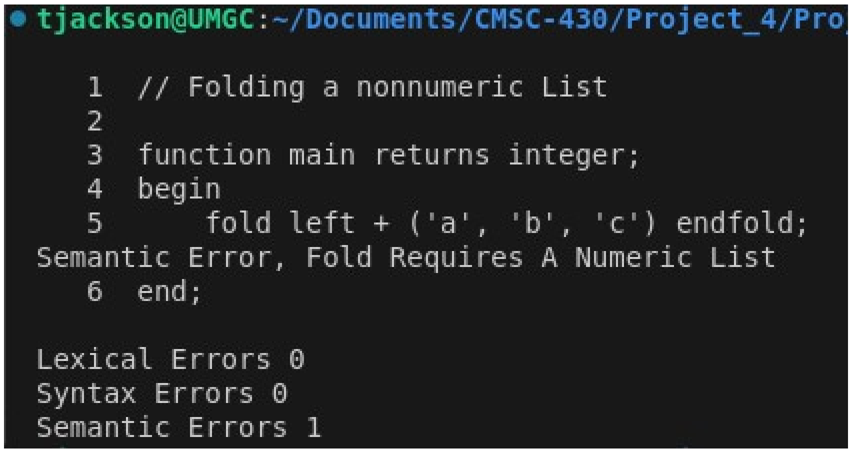
```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/P

-- If Elsif Else Mismatch

function main returns integer;
begin
    if 9 < 10 then
        1;
    elsif 8 = 1 then
        2;
    else
        3.7;
    endif;
Semantic Error, If-Elsif-Else Type Mismatch
end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 15:



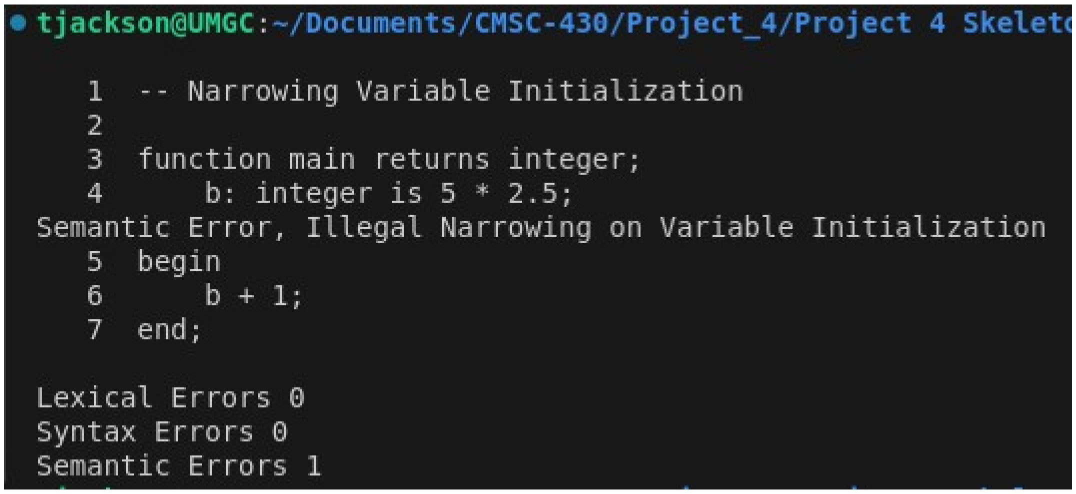
```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/Pro

// Folding a nonnumeric List

function main returns integer;
begin
    fold left + ('a', 'b', 'c') endfold;
Semantic Error, Fold Requires A Numeric List
end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 1
```

Semantic 16:



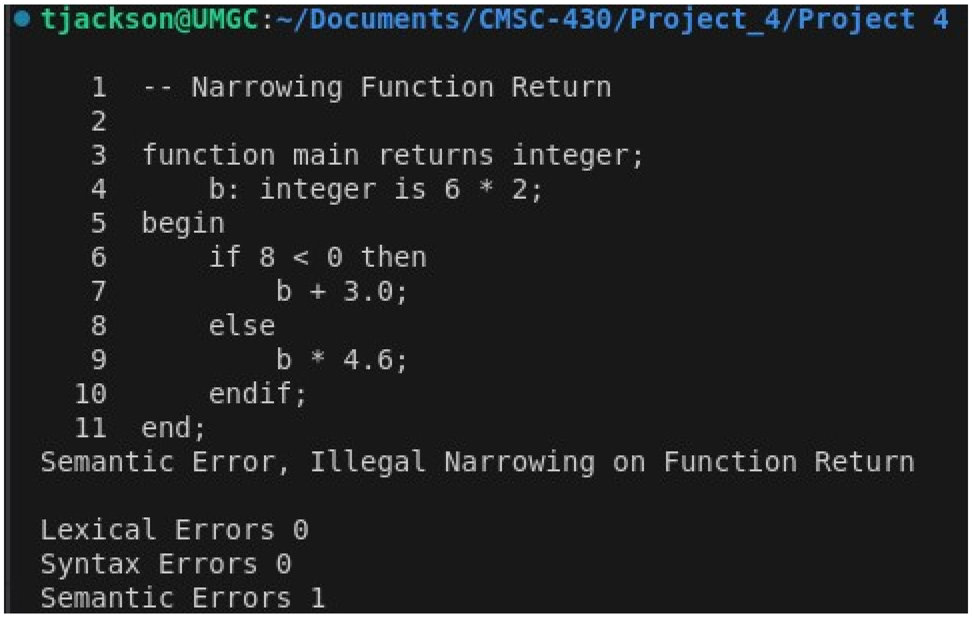
```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project 4 Skelet

   -- Narrowing Variable Initialization

   function main returns integer;
       b: integer is 5 * 2.5;
 Semantic Error, Illegal Narrowing on Variable Initialization
   begin
       b + 1;
   end;

 Lexical Errors 0
 Syntax Errors 0
 Semantic Errors 1
```

Semantic 17:



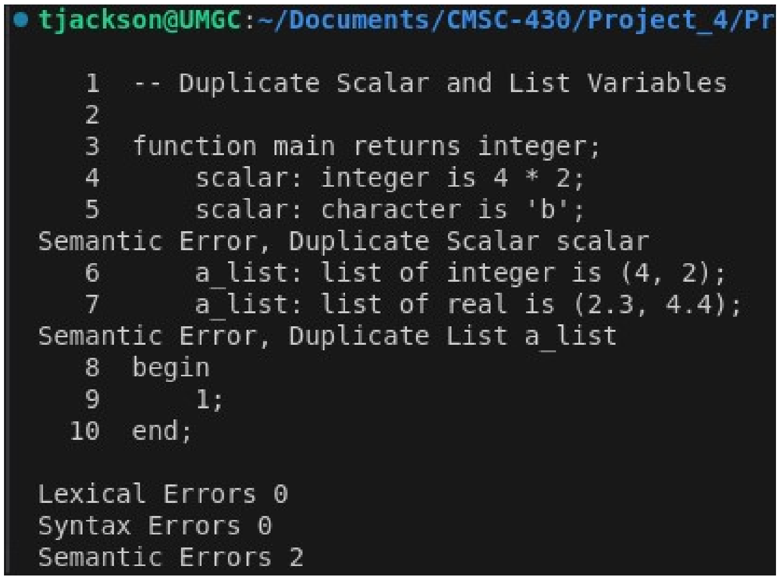
```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project 4

   -- Narrowing Function Return

   function main returns integer;
       b: integer is 6 * 2;
   begin
       if 8 < 0 then
           b + 3.0;
       else
           b * 4.6;
       endif;
   end;
 Semantic Error, Illegal Narrowing on Function Return

 Lexical Errors 0
 Syntax Errors 0
 Semantic Errors 1
```

Semantic 18:



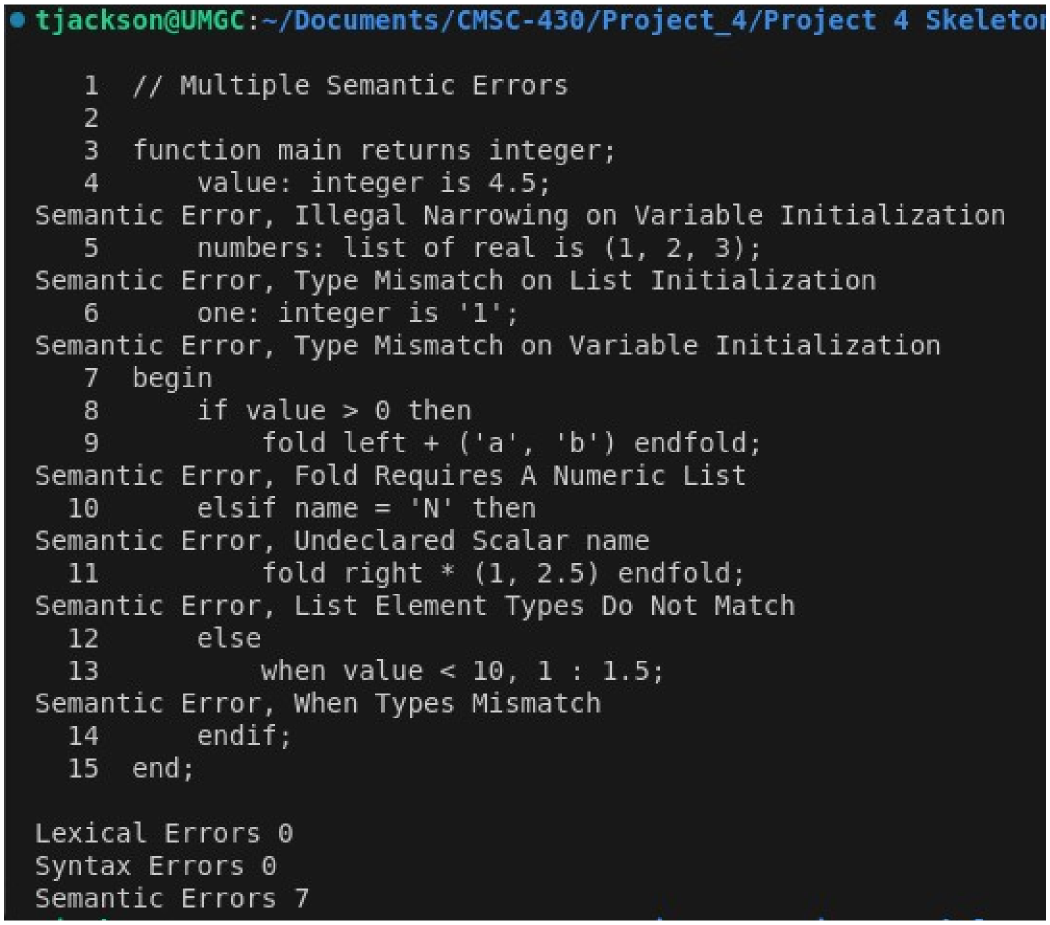
```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/Pr

-- Duplicate Scalar and List Variables

function main returns integer;
    scalar: integer is 4 * 2;
    scalar: character is 'b';
Semantic Error, Duplicate Scalar scalar
    a_list: list of integer is (4, 2);
    a_list: list of real is (2.3, 4.4);
Semantic Error, Duplicate List a_list
begin
    1;
end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 2
```

Semantic 19:



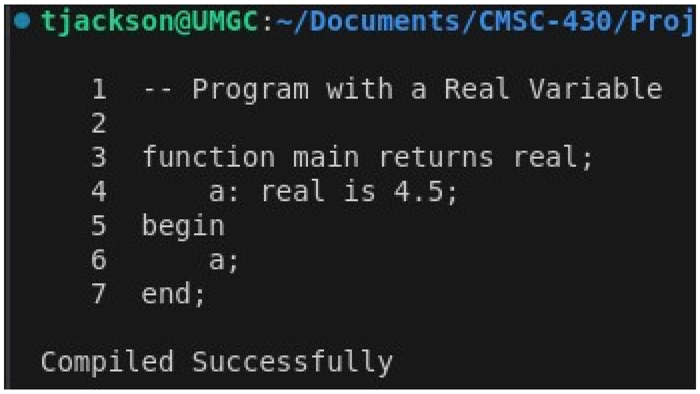
```
tjackson@UMGC:~/Documents/CMSC-430/Project_4/Project 4 Skeleton

// Multiple Semantic Errors

function main returns integer;
    value: integer is 4.5;
Semantic Error, Illegal Narrowing on Variable Initialization
    numbers: list of real is (1, 2, 3);
Semantic Error, Type Mismatch on List Initialization
    one: integer is '1';
Semantic Error, Type Mismatch on Variable Initialization
begin
    if value > 0 then
        fold left + ('a', 'b') endfold;
Semantic Error, Fold Requires A Numeric List
    elsif name = 'N' then
Semantic Error, Undeclared Scalar name
        fold right * (1, 2.5) endfold;
Semantic Error, List Element Types Do Not Match
    else
        when value < 10, 1 : 1.5;
Semantic Error, When Types Mismatch
    endif;
end;

Lexical Errors 0
Syntax Errors 0
Semantic Errors 7
```

Valid 1:



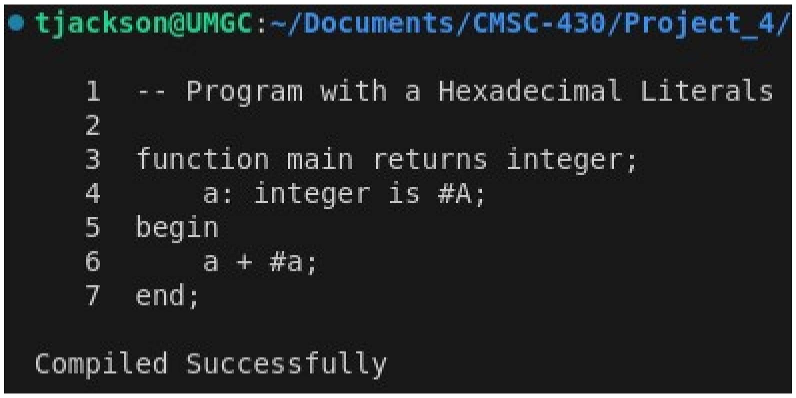
```
● tjackson@UMGC:~/Documents/CMSC-430/Proj

    -- Program with a Real Variable

    function main returns real;
        a: real is 4.5;
    begin
        a;
    end;

 Compiled Successfully
```

Valid 2:



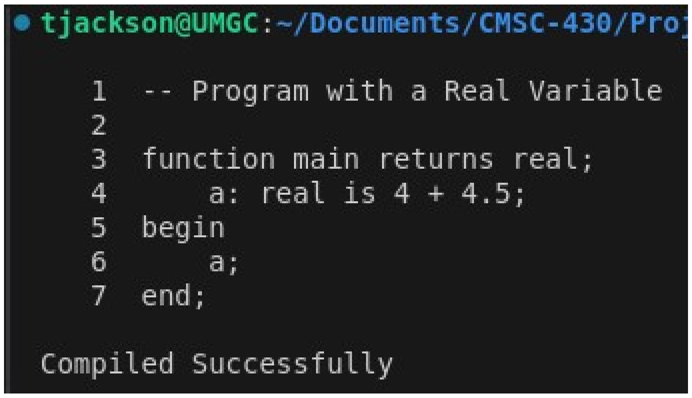
```
● tjackson@UMGC:~/Documents/CMSC-430/Project_4/

    -- Program with a Hexadecimal Literals

    function main returns integer;
        a: integer is #A;
    begin
        a + #a;
    end;

 Compiled Successfully
```

Valid 3:



```
● tjackson@UMGC:~/Documents/CMSC-430/Proj

    -- Program with a Real Variable

    function main returns real;
        a: real is 4 + 4.5;
    begin
        a;
    end;

 Compiled Successfully
```

Lessons Learned

This project highlighted how useful it can be to reuse and adapt existing code rather than creating new functions from scratch every time. Doing this saved time and reduced redundancy, particularly in handling variable initialization and numeric validation. Additionally, I realized how interconnected various components of the semantic analyzer are. In working with relational operators, I found potential impacts on arithmetic operators. Modifying one part of the analyzer often had ripple effects on other areas, reinforcing the need for a holistic approach when making changes. Finally, troubleshooting the bug in my checkIf() function highlighted the importance of thorough testing. It’s crucial to consider edge cases—such as if statements without elsif branches—that might not be immediately obvious but could cause errors later on.