CMSC 430 Project 4

The fourth project involves modifying the semantic analyzer for the attached compiler by adding checks for semantic errors. The static semantic rules of this language are the following:

Variable and parameter names have local scope. The scope rules require that all names be declared and prohibit duplicate names within the same scope. The type correspondence rules are as follows:

- Both expressions in the when statement must be the same type
- The type of the `switch` expression must be Integer.
- All the `case` statements in a `switch` statement must match in type. No coercion is performed.
- Arithmetic operators can only be used with numeric types.
- All list elements must be of the same type.
- In a list variable declaration, the type of the list must match the type of the elements.
- List subscripts must be integer expressions.
- Character literals can be compared to one another but they cannot be compared to numeric expressions.
- Only integer operands can be used with the remainder operator.
- All the statements in an `if` statement must match in type. No coercion is performed.
- In a `fold` statement, the list must be a numeric type.
- A narrowing variable initialization or function return error occurs when a real value is being forced into integer. Widening is permitted. In all other cases, the type must match.

Type coercion from an integer to a real type is performed within arithmetic expressions.

You must make the following semantic checks. Those highlighted in yellow are already performed by the code that you have been provided, although you are must make minor modifications to account for the addition of real types and the need to perform type coercion and to handle the additional arithmetic, relational and logical operators.

- Type Mismatch on Variable Initialization
- When Types Mismatch
- Switch Expression Not Integer
- Case Types Mismatch
- Arithmetic Operator Requires Numeric Types
- Undeclared Scalar
- Undeclared List
- List Element Types Do Not Match
- List Type Does Not Match Element Types
- List Subscript Must Be Integer
- Character Literals Cannot be Compared to Numeric Expressions
- Remainder Operator Requires Integer Operands

- If-Elsif-Else Type Mismatch
- Fold Requires A Numeric List
- Illegal Narrowing Variable Initialization
- Illegal Narrowing Function Return
- Type Mismatch on Function Return
- Duplicate Scalar
- Duplicate List

This project requires modification to the bison input file, so that it defines the additional semantic checks necessary to produce these errors and addition of functions to the library of type checking functions already provided in `types.cc`. You must also make some modifications to the functions provided. You need to add a check to the `checkAssignment` function for mismatched types in the case that nonnumeric and numeric types are mixed. You need to also add code to the `checkArithmetic` function to coerce integers to reals when the types are mixed.

The provided code includes a template class `Symbols` that defines the symbol table. It already includes a check for undeclared identifiers. You need to add a check for duplicate identifiers.

Like the lexical and syntax errors, the compiler should display the semantic errors in the compilation listing, after the line in which they occur. An example of a compilation listing output containing semantic errors is shown below:

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

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

You are to submit two files.

1. The first is a `.zip` file that contains all the source code for the project. The `.zip` file should contain the flex input file, which should be a `.l` file, the bison file, which should be a `.y` file, all `.cc` and `.h` files and a `makefile` that builds the project.
2. The second is a Word document (PDF or RTF is also acceptable) that contains the documentation for the project, which should include the following:
    a. A discussion of how you approached the project
    b. A test plan that includes test cases that you have created indicating what aspects of the program each one is testing and a screen shot of your compiler run on that test case
    c. A discussion of lessons learned from the project and any improvements that could be made