San José State University
Department of Computer Engineering

CMPE 152
Compiler Design

Fall 2018
Instructor: Ron Mak

Assignment #4

Assigned: Thursday, September 27
Due: Monday, October 8 at 11:59 pm
Team assignment, 100 points max

New built-in complex type

Add a new built-in **`complex`** type to Pascal for performing complex arithmetic, i.e., with imaginary numbers. This type should be a record type with two **`real`** fields named **`re`** and **`im`** for the real and imaginary parts, respectively, of a **`complex`** number.

Parse and execute complex assignments

You should be able to parse declarations of **`complex`** variables and assignments to them using the **`re`** and **`im`** fields, and then execute those statements.

After you implement the new built-in **`complex`** type, you should be able to parse and execute the test input file **`ComplexAssignments.txt`**

http://www.cs.sjsu.edu/~mak/CMPE152/assignments/4/ComplexAssignments.txt

```
VAR
    x, y, z : complex;

BEGIN
    x.re := 15;
    x.im := 37;

    y.re := -12.34;
    y.im := 3.1415926;

    z := x;
END.
```

Turn on the cross-reference listing and the parse tree listing with the **`-ix`** command-line options.

Procedure

Start with the C++ source files from Chapter 10.

Examine **`wci::intermediate::symtabimpl::Predefined`** to see how the built-in types like **`integer`** and **`real`** are created and entered into the global symbol table.

Examine **`wci::frontend::pascal::parsers::RecordTypeParser`** to see how it creates a symbol table for the record type and enters the fields into the record type's symbol table.

In **`wci::intermediate::symtabimpl::Predefined`**, create the new built-in complex record type and enter two real fields, **`im`** and **`re`**, into its symbol table.

> The only files you should need to change from the Chapter 10 source files are **Predefined.h** and **Predefined.cpp**.

Once you've defined the built-in **`Complex`** type as a record type with fields **`re`** and **`im`**, the assignment statements should "just work", since they ought to be no different from any other assignments to record fields.

Execute the test input file. The debugging output from each assignment statement should indicate that the assignment statements execute properly.

What to turn in

This is a team assignment. Each team turns in one assignment and each team member will get the same score. Create a zip file that contains:

- All your **`.h`** and **`.cpp`** source files.
- A text file that contains the listing, cross-reference, parse trees and execution output from compiling and executing the test input file with the new built-in **`complex`** type.

Submit into Canvas: **Assignment #4: New Complex Type**.

Rubric

Your program will be graded according to these criteria:

Criteria	Max points
• Good modifications to compiler source files **Predefined.h** and **Predefined.cpp**. • Successfully parse test input file **ComplexAssignments.txt**. • Correct cross-reference and parse trees from compiling the test input file. • Good execution output.	• 25 • 25 • 25 • 25