San José State University
Department of Computer Engineering

CMPE 152
Compiler Design

Sections 4 and 5
Spring 2018
Instructor: Ron Mak

Assignment #4

Assigned: Tuesday, February 27
Due: Wednesday, March 7 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.

You should be able to declare **`complex`** variables like scalars. For example:

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

Parse `complex` assignments

To assign a value to a **`complex`** variable, use the **`re`** and **`im`** fields. For example:

```
z.re := 3.14;
z.im := -8.2;
```

After you implement the new built-in **complex** type, your new Pascal compiler and interpreter will be able to parse the test source 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.

Tips

Start with the C++ source code of Chapter 10. (Starting with Chapter 9 will only allow you to declare complex variables. Starting with Chapter 10 will enable the assignments to complex variables.)

Examine **wci::intermediate::symtabimpl::Predefined** to see how the built-in types like **integer** and **real** are defined.

Examine **wci::frontend::pascal::parsers::RecordTypeParser** to see what information is entered into the symbol table for a **record** type.

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.

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. Verify that Chapter 10's **`makefile`** can compile source file **`ComplexAssignments.txt`**.

```
make compile src=ComplexAssignments.txt
```

- A text file that contains the listings and output from compiling the above test source 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
• Correct entry for **`complex`** in the global symbol table for the new built-in type. • Parser modifications to handle the new **`complex`** type. • Successfully parse program **`ComplexAssignments.txt`**. • Correct parse trees for **`complex`** assignment statements.	• 25 • 25 • 25 • 25