# EECS665

## Compiler Construction

Drew Davidson
Ruturaj Vaidya

Lecture: LEEP2 G415
MWF 3:00-3:50

Lab: Eaton 1005B

ANOUNCEMENTS | LAB | SCHEDULE | MATERIALS | ASSIGNMENTS

## Homework 6

Due on October 17th @ 3:00 PM (in class, to Drew, or at Engineering front office)

Not accepted late

ALL homework must be done individually

# Questions

The C language allows you to define new names for existing types using typedefs. Here is some example code that uses typedefs:

```
typedef int money;
int x;
money y;
typedef money dollars;
dollars z;
```

```
x = 10;
y = x;    // OK because x and y are of type int
z = y;    // OK because y and z are of type int
```

The first typedef defines money to be a synonym for int. Any declaration that follows this typedef can use money instead of int. The second typedef defines dollars to be a synonym for money, which makes it a synonym for int. Any declaration that follows this typedef can use dollars instead of int.

Typedefs can also be used with struct types:

```
struct Pair {
    int x;
    int y;
};
typedef struct Pair Point;
Point p;
```

A typedef can occur anywhere that a variable declaration (local or global) can occur. The usual C scoping rules apply to the names in typedefs. Note that typedef int money; is considered to be a declaration of the name money and that both money x; and typedef money dollars; are

considered to be uses of the name money.

# Question 1

Assume that the following productions have been added to the grammar for the Lil' C language:

```
decl → typedef
typedef → TYPEDEF type ID SEMICOLON
```

What other productions need to be changed and/or added to the Lil' C grammar to allow typedefs?

# Question 2

# Question 2

Now consider the name-analysis phase of the compiler. Note that, in addition to the usual errors for multiply-defined names and for uses of undefined names, the name analyzer must enforce the following rules:

- The declaration typedef T xxx; is an error if T is not a built-in type (e.g., int, bool) or a struct type (in which case T will be of the form: struct ttt) or a new type defined by a previous typedef in the current or an enclosing scope.
- The declaration typedef T xxx; is an error if xxx has already been declared in the current scope (as a variable, function, parameter, or new type).
- A variable, function, or parameter can only be declared to be of type T if T is either a built-in type or a new type defined by a previous typedef in the current or an enclosing scope. (A variable can still be declared to be of type struct ttt as before.)

Answer each of the following questions:

- What information should be stored with each name in the symbol table?
- What should be done to process a typedef: typedef T xxx;?
- What should be done to process a declaration of a variable, function, or parameter named xxx and declared to be of type T?
- What should be done to process the use of a name xxx in a statement?

Illustrate your answer by showing the entries that would be in the symbol table after processing the following declarations:

```
struct MonthDayYear {
    int month;
    int day;
    int year;
};
typedef struct MonthDayYear date;
date today;
typedef int dollars;
dollars salary;
typedef dollars moreDollars;
moreDollars md;
int d;
```