TOOL, mini-project:

Improved Arguments and Fields

Improved Arguments and Fields

- Implement named arguments with potential default values. Allow default values to class members as well. The default values can be arbitrary Tool expressions.

- Implement a mechanism for providing an arbitrary number of arguments to a function.

- Provide Tool classes with an extra copy() method that allows a quick cloning of an object while specifying which field to modify (Similar to what Scala's copy() for case classes)

Implement named arguments with potential default values. Allow default values to class members as well. The default values can be arbitrary Tool expressions.

We will allow scala-like default values for parameters, a typical usage is:

```
def foo(bar: Int = 1000, fb: String = "SSL"): Type = { … }
```

Implement a mechanism for providing an arbitrary number of arguments to a function.

We will modify our grammar as follows:

def *Identifier* **(** (*Identifier* **:** *Type(*)* (**,** *Identifier* **:** *Type(*)*)*)? **)** **:** *Type* **= {** (*VarDeclaration*)* (*Statement*)* **return** *Expression* **}**

In other words, we permit the following scala-like method declaration:

def foo(bar: String*): Type = { … }

To implement this new feature we will mainly modify the lexer, parser.

We will maybe only allow a single starred parameter per function.

Provide Tool classes with an extra copy() method that allows a quick cloning of an object while specifying which field to modify (Similar to what Scala's copy() for case classes)

Again, we will provide scala-like copy() method.

A typical usage is:

class Person { var name: String; var age: Int; def setName(...) {} … }

var alice: Person; alice = new Person(); alice.setName(“alice”); …

var bob: Person; bob = new Person(); bob = alice.copy(name = “bob”); ...