COMP 410
Summer 2024

Nondeterminism in Scala

1.) Consider the following Prolog procedure:

```
myNumber(0).
myNumber(1).
myNumber(2).
```

Write an equivalent generator function in Scala. The signature is below:

```
def myNumber: Iterator[Int] =
```

2.) Consider the following Prolog procedure, which uses `myNumber`:

```
makePair(pair(A, B)) :-
  myNumber(A),
  myNumber(B).
```

Write an equivalent generator function in Scala. It should generate Scala pairs instead. For example, the Prolog `pair(1, 2)` is equivalent to the Scala `(1, 2)`. The signature is below:

```
def makePair: Iterator[(Int, Int)] =
```

3.) Consider the following Prolog procedure, representing a generator for an AST:

```
gen(_, variable(x)).
gen(_, variable(y)).
gen(Depth, call(E1, E2)) :-
  Depth > 0,
  NewDepth is Depth - 1,
  gen(NewDepth, E1),
  gen(NewDepth, E2).
```

Assume we have the following Scala code defined, representing the AST:

```
sealed trait Exp
case class Variable(name: String) extends Exp
case class Call(e1: Exp, e2: Exp) extends Exp
```

Write an equivalent generator in Scala. You can assume the variable names are strings (e.g., "x", "y"). The signature is below:

```
def gen(depth: Int): Iterator[Exp] =
```