COMP 410
Fall 2023

Final Exam Review with Answers

For the final, all handouts, labs, practice exams, and prior exams are fair game. You may bring three 8.5 x 11 inch sheets of paper into the exam, and have handwritten notes on both sides of both sheets. This review focuses on performing logic programming in Python.

1.) Consider the following Prolog procedure:

```
isName(alice).
isName(bob).
isName(janet).
isName(bill).
```

Write an equivalent generator function in Python, named `isName`. Each name should be represented as a string. As a hint, `isName` should not take any parameters.

```
def isName():
  yield "alice"
  yield "bob"
  yield "janet"
  yield "bill"
```

2.) Consider the following Prolog procedure:

```
naturalNumber(0).
naturalNumber(N) :-
  naturalNumber(NMinusOne),
  N is NMinusOne + 1.
```

Write an equivalent generator function in Python, named `naturalNumber`. As a hint, `naturalNumber` should not take any parameters.

```
def naturalNumber():
  yield 0
  for nMinusOne in naturalNumber():
    yield nMinusOne + 1
```

3.) Consider the following Prolog procedure:

```
selectElement([Head|_], Head).
selectElement([_|Tail], Element) :-
    selectElement(Tail, Element).
```

Write an equivalent generator function in Python, named `selectElement`. You can assume you have the following definitions available for representing lists:

```
class Nil:
    def __init__(self):
        pass

class Cons:
    def __init__(self, head, tail):
        self.head = head
        self.tail = tail
```

Example usage of `selectElement` is below:

```
for n in selectElement(Cons(1, Cons(2, Cons(3, Nil())))):
  print(n)

# Output:
# 1
# 2
# 3
```

```
def selectElement(inputList):
    if isinstance(inputList, Cons):
     yield inputList.head
        for element in selectElement(inputList.tail):
            yield element
```

4.) Consider the following Prolog procedure, which nondeterministically selects different values contained in a binary tree:

```
% tree ::= internal | node(tree, INT, tree)
treeElement(node(_, Value, _), Value).
treeElement(node(Left, _, _), Value) :-
    treeElement(Left, Value).
treeElement(node(_, _, Right), Value) :-
    treeElement(Right, Value).
```

The tree is represented in Python using the following two classes:

```
class Internal:
    def __init__(self):
        pass

class Node:
    def __init__(self, left, value, right):
        self.left = left
        self.value = value
        self.right = right
```

Write an equivalent generator function implementing `treeElement` in Python below.

```
def treeElement(node):
    if isinstance(node, Node):
     yield node.value
        for value in treeElement(node.left):
            yield value
        for value in treeElement(node.right):
            yield value
```