CSE 505 Fall 2015

Assignment 5: Lambda Calculus and Logic Programming

Assigned: Mon, Nov 23, 2015

Due: Mon, Dec 7 (11:59 pm)

Note: This assignment may be done by a pair of students.

Problem 1. Consider the representation of a **queue** of n elements e1 … e_n in the lambda-calculus:

λc. λn.((c e_1) ((c e_2) … ((c e_n) n) …)).

Show *non-recursive* lambda-calculus definitions for the following two operations on a queue:

(i) `insert` – given a queue `q` and element `e`, return a queue by adding the element `e` at the end of q.

(ii) `length` – given a queue q, return a Church numeral denoting the number of elements in `q`.

Examples:

```
((insert tom) λc.λn.n)                    (length λc.λn.n)
     =>* λc.λn.((c tom) n)                        =>* λf.λx.x
((insert ding) λc.λn.((c tom) n))         (length λc.λn.((c tom) ((c ding) n)))
     =>* λc.λn.((c tom) ((c ding) n))             =>* λf.λx.(f (f x))

((insert hari) λc.λn.((c tom) ((c ding) n)))
     =>* λc.λn.((c tom) ((c ding) ((c hari) n))))
(length λc.λn.((c tom) ((c ding) ((c hari) n))))
     =>* λf.λx.(f (f (f x)))
```

Save your definition for `insert` and `length` in a file called `church.txt`. It is acceptable to write `L` instead of `λ` in your definitions for `insert` and `length`.

Problem 2.

(a) Consider a representation for a lambda-term using three Prolog constructors `v, l,` and `a,` for variable, abstraction, and application respectively. For example, a lambda-term ((λf.λx.(f x) y) z) would be represented by the Prolog term

```
a(a(l(f, l(x, a(v(f), v(x)))), v(y)), v(z)).
```

Write a Prolog predicate `reducible(T)`, which returns `true` if T has a beta- or eta-redex somewhere inside it; otherwise, the predicate returns `false`. Examples:

```
?- reducible(l(x,v(x))).          ?- reducible(a(l(x,v(x)), v(y))).
     false                             true

?- reducible(a(a(l(f, l(x, a(v(f), v(x)))), v(y)), v(z))).
     true
```

(b) Using `reducible`, write a Prolog predicate called `norm(T)` which returns `true` if T is in normal form; otherwise, the predicate returns `false`.

Save your answers to parts (a) and (b) in a file called `lambda.pl`.

Problem 3. Four boys (Ali, Bing, Charles, Dani) and four girls (Kari, Lola, Mary, Nina) need to be assigned to one of four activities (biking, running, hiking, surfing). Each boy and girl should be assigned to only one activity each and their individual constraints are as follows:

a. Ali likes to bike and Mary likes to hike.
b. Bing, Charles, Kari and Lola do not like to run.
c. Nina does not like to surf.
d. Lola and Charles want to be together.
e. Dani and Mary do not want to be together.

Write a Prolog program to assign one boy and one girl for each activity, as follows:

- The boys and girls should be defined by two predicates:

```
boy(ali).                    girl(kari).
boy(bing).                   girl(lola).
boy(charles).                girl(mary).
boy(dani).                   girl(nina).
```

- Each activity should be defined as a binary constructor: `biking(Boy1,Girl1)`, `running(Boy2,Girl2)`, `hiking(Boy3,Girl3)`, and `surfing(Boy4,Girl4)`.
-
- The top-level predicate should be called `solve` and it should be defined like this:

```
solve(Answer) :- assumptions(Answer),
                 constrainta(Answer),  constraintb(Answer),
                 constraintc(Answer),  constraintd(Answer),
                 constrainte(Answer).

assumptions(Answer) :- boy(B1), boy(B2), boy(B3), boy(B4),
                       girl(G1), girl(G2), girl(G3), girl(G4),
                       Answer = [biking(B1, G1), running(B2, G2),
                                 hiking(B3, G3), surfing(B4, G4)].
```

- Each of the constraints a – e should be defined using one Prolog rule and *should incorporate only the conditions stated in that constraint*. To help you get started, we can define

```
constrainta(Answer) :-  member(biking(ali,_), Answer),
                        member(hiking(_,mary), Answer).
```

Develop Prolog predicates for the constraints b – e and place the code for solve, assumptions, and constrainta … constrainte in a file **assign.pl**. Note: The inequality operator in Prolog is `\==`, and is used, for example, as `B \== charles`.

What to Submit: Prepare a top-level directory named *A5_UBITId1_UBITId2* if the assignment is done by two students; otherwise, name it as *A5_UBITId* if the assignment is done solo. (Order the *UBITId*'s in alphabetic order, in the former case.) In this directory, place **church.txt, lambda.pl,** and **assign.pl.** Compress the directory and submit the resulting compressed file using the `submit_cse505` command. For more details regarding online submission, see `Resources` → `Homeworks` → `Online_Submission_2015.pdf`.

End of Assignment #5