CSE 505 Spring 2017

Assignment 6: Lambda Calculus and Logic Programming

Assigned: Sat, April 29, 2017

Due: Thurs May 11, 2017 (11:59 pm)

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

Problem 1. Consider a **list** of n elements e1 … e_n represented in the lambda-calculus as:

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 list:

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

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

Examples:

```
((insert tom) Lc.Ln.n)
     =>* Lc.Ln.((c tom) n)
((insert hari) Lc.Ln.((c tom) ((c ding) n)))
     =>* Lc.Ln.((c tom) ((c ding) ((c hari) n))))
(length Lc.Ln.n)
      =>* Lf.Lx.x
 (length Lc.Ln.((c tom) ((c ding) ((c hari) n))))
     =>* Lf.Lx.(f (f (f x)))
```

Save your definitions for `insert` and `length` in a file called `problem1.txt`. (You may want to test your definition using the lambda-calculus simulator posted on Piazza.)

Problem 2. Consider a representation for a lambda-term using two Prolog constructors `l` and `a` which stand for λ-abstraction and application respectively. For example, ((λf.λx.(f x) y) z) would be represented by the Prolog term

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

(a) Write a Prolog predicate `church(N, T)` which given a non-negative integer N returns in T a Prolog term representing the church numeral representation for N. Examples:

```
?- church(0, T)
     l(f, l(x, x))

?- church(3, T)
    l(f, l(x, a(f, a(f, a(f, x)))))
```

(b) Given a Prolog term T representing a λ-term, write a predicate `pretty_print(T)` which prints out T as a string following λ-calculus syntax (using L for λ). See Problem 1 for sample strings. Note: The built-in predicate `write(S)` prints out string S, e.g., write('(hello)'). In Prolog, a string is enclosed in a pair *single quotes*.

Problem 3. Consider the predicate `norm(T)` defined below for computing the normal form of an input λ-term T by repeatedly performing a one-step reduction.

```
norm(T) :-
     write('--> '),
     pretty_print(T), nl,
     reduce(T, R),
     (T == R              % if
          -> true         % then
          ; norm(R)).     % else
```

Define the predicate `reduce(T, R)` which performs a one-step reduction on T to derive `R`. If there is more than one β- or η-redex in T, the leftmost redex should be chosen for reduction. If there is no redex in T, the term R should be equal to T.

The definition of `reduce(T, R)` is a case analysis on the structure of input term T and can be stated in nine cases, the first of which is:

```
reduce(X, X) :-
     atom(X).    % X is a variable
```

There are five cases when T is a λ-abstraction, one of which corresponds to the case when T is an η-redex. There are three cases when T is an application one of which corresponds to the case when T is a β-redex.

(Note: It is not necessary that you should have exactly nine cases for reduce. A correct definition that has more or fewer cases is perfectly acceptable.)

The outline of your program is given on Piazza in the file Resources → Homeworks → `lambda.pl`. Complete the program and check its correctness using the test cases given in the file. The answers to the test cases are also posted on Piazza.

WHAT TO SUBMIT:

Prepare a top-level directory named *A6_UBITId1_UBITId2* if the assignment is done by two students; otherwise, name it as *A6_UBITId* if the assignment is done solo. (Order the *UBITId*'s in alphabetic order, in the former case.) In this directory, place

problem1.txt and **lambda.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_2017.pdf`.

End of Assignment #6