CSE340 Fall 2019 HOMEWORK 3

Due Wednesday 24 October 2019 by 12:01 AM

PLEASE READ THE FOLLOWING CAREFULLY

1. Your answers can be handwritten or typed. If you write your answers, you need to have clear handwriting. If we cannot read it we cannot grade it!
2. On gradescope, you should submit the answers to separate question separately.

Problem 1 (Lambda calculus normal and applicative order). For each of the following identify the redex (if any) that will be reduced first under normal order reduction strategy. You are only asked to identify the redex not to do the reduction.

a. x (λx. x x) (λx. x x) x

b. x (λx. (λx. x) x) (λx. x) x

c . (λx. (λx. x) x) λx. x

d. ((λx. λy. (λy. x) λx. x x)) ((λx. (λy. x)) λx. x x)

e. (λx. (λx. (λx. x) x) x) (x (λx. x) x)

f. x (λx. (λy. x) λx. x x) (λx. (λy. x) λx. x x) ((λx. (λy. x)) λx. x x)

Question 2 (Lambda calculus normal and applicative order). For each of the following expressions identify a redex (if any) that can be reduced first under call by value reduction strategy. Remember that call by value does not specify which redex should be reduced first. It specifies which redex cannot be reduced first (a redex whose right hand side is not a value) or cannot be reduced (under abstraction). If there is more than one redex that can be reduced first, you should specify all of them.

a. x (λx. x x) (λx. x x) x

b. x (λx. (λx. x) x) (λx. (λx. x) x)

c . (λx. (λx. x) x) (λx. (λx. x) x)

d. (λx. (λy. x) λx. x x) ((λx. (λy. x)) λx. x x)

e. (λx. (λx. (λx. x) x) x) (x (λx. x) x) ((λx. (λx. (λx. x) x)) x)

Question 3 (Linked lists with lambda calculus). We define a lambda calculus representation of linked lists. A list is represented with pairs. The first part of the pair is an element and the second part of the pair is a list. In order to indicate if the second element represents a non-empty list, we use a boolean flag. We also, use a header in order to be able to represent the empty list. I start by giving a few examples.

empty list = pair fls fls

list with one element a = pair tru (pair a (pair fls fls))

representation of empty list

list with two elements a and b = pair tru (pair a (pair tru (pair b (pair fls fls))))

representation of list with one element b

In general, the representation of a list with elements $a_1, a_2, a_3, \ldots, a_k$ in the given order, where **k ≥ 1** is

pair tru (pair a_1 L)

where L is the representation of the list whose element in order are $a_2, a_3, \ldots, a_k$

Note that only the last pair has "fls" as the first element to indicate the empty list. For all the questions, you are asked to write functions. You should understand that to mean write a lambda expression.

1. (**Add element**) Write a function that takes an element a and a list L representing $a_1, a_2, a_3, \ldots, a_k$ and returns a list representing $a, a_1, a_2, a_3, \ldots, a_k$
2. (**Empty List**) Write a function that returns a boolean value to indicate if a list if empty or not. The function should return tru if the list is empty and fls if the list is not empty.
3. (**First Element**) Write a function that returns the first element of a list. Since a list can be empty, we need a way to indicate that. Your function should return a pair. If the first element of the pair is tru, then the second element if the first element of the list. If the first element of the pair is false, this means that the list is empty and has no first element.
4. (**Last element**) Write a recursive function that returns the last element of a list. Again as in question 3 , the function should return a pair. The first part of the pair is a boolean value. If the first element of the pair is tru, then the second element if the first element of the list. If the first element of the pair is false, this means that the list is empty and has no first element.
5. (**Sum of elements**) Write a recursive functions that takes a list of integers as an argument and returns the sum of the elements of the list. If the list is empty, the function should return 0
6. (**Reversing a List**) Write a recursive function that reverses a list. If the list represents $a_1, a_2, a_3, \ldots, a_k$, the reverse of the list should represent $a_k, a_{k-1}, a_{k-2}, \ldots, a_1$
7. (**Merging two lists**) Write a recursive function that takes two lists L1 and L2 and returns a list L obtained by appending L2 to L1. If L1 represents represents $a_1, a_2, a_3, \ldots, a_m$ and L2 represents $b_1, b_2, b_3, \ldots, b_n$, then the result should represent $a_1, a_2, a_3, \ldots, a_m, b_1, b_2, b_3, \ldots, b_n$.

Question 4 (scoping). Give the output of the following program under

1. static scoping
2. dynamic scoping

```
int a = 5;
int b = 6;

void f()
{

   g();

   { int a = 4;
     a = a + b;
     b = a + b;
     g();
   }

   { int b = 3;
     a = a + b;
     b = a + b;
     g();
   }

   a = a+b;
   g();

}

void g()
{
   print a;
   print b;
}

int main ()
{
    int a = 1;
    int b = 10;
    f();
    g();
}
```

Show your work!