CSE340 Spring 2020 HOMEWORK 4
Due by 11:59 PM on Thursday April 2 2020

PLEASE READ THE FOLLOWING CAREFULLY

1. Your answers for questions 1 and 2 must be typed.
2. Your answers to question 3 can be handwritten but it has to be neat and legible. Your answers to question 3 can also be typed if you prefer.
3. On Gradescope, you should submit the answers to separate question separately.
4. For each question, read carefully the required format for the answer. The required format will make it easier for you to answer and for the graders to grade. Answers that are not according to the required format will not be graded

Problem 1 (Lambda Calculus). The goal of this problem is to give you further practice with lambda calculus. Each part of this problem will have an expression that you are asked to evaluate or simplify as much as possible. The following are some examples

Example 1. plus2 = λn. succ (succ n)
what does the following evaluate to: 4 plus2 2
Answer. 10

Example 2. quad = λx. λy. λz. λw. pair (pair x y) (pair z w)
what does the following evaluate to: succ (fst (snd (quad 1 3 5 7)))
Answer. 6

We will use the following definitions in what follows

next1 = λp. pair (times (fst p) (snd p)) (succ (snd p))

next2 = λp. pair (snd p) (fst p) (**note**: there were extra parentheses that I removed)

next3 = λn. (times n n)

For each of the following, give the value that the expressions evaluates to

1. what is next1 (pair 1 1)?

   next1 (pair 1 1) = (λp. pair (times (fst p) (snd p)) (succ (snd p))) (pair 1 1) =

   = pair (times (fst (pair 1 1)) (snd (pair 1 1))) (succ (snd (pair 1 1))))

   = pair (times 1 1) (succ 1)

   = pair 1 2

2. what is next1 (next1 (pair 1 1))?

   next1 (next1 (pair 1 1)) = next1 (pair 1 2)

   = (λp. pair (times (fst p) (snd p)) (succ (snd p))) (pair 1 2) =

   = pair (times (fst (pair 1 2)) (snd (pair 1 2))) (succ (snd (pair 1 2))))

= pair (times 1 2) (succ 2)

= pair 2 3

3. what does the function λn. fst (n next1 (pair 1 1)) calculate?
    - If we calculate next1 (next1 (next1 (pair 1 1))), we get pair 6 4
    - If we calculate next1(next1 (next1 (next1 (pair 1 1)))), we get pair 30 5
    - Applying next1 n times starting with (pair 1 1), we get (pair n! (n+1)) (I am using familiar notation to make it clearer).

    The function we are given applies n to next1 and (pair 1 1) and then takes the fst element of the resulting pair. So, we get next1 applied n times to (pair 1 1) = (pair n! (n+1)), whose fst element is n!. So, the function calculated n!
4. what is fst (next2 (pair tru fls))?

    fst (next2 (pair tru fls)) = fst ((λp. pair (snd p) (fst p)) (pair tru fls))

    = fst (pair (snd (pair tru fls)) (fst (pair tru fls)))

    = fst (pair fls tru)

    = fls
5. what is fst (next2 (next2 (pair tru fls)))?

    fst (next2 (next2 (pair tru fls))) = fst (next2 (pair fls tru)) // derivation similar to above

    = fst (pair (snd (pair fls tru)) (fst (pair fls tru)))

    = fst (pair tru fls)

    = tru

6. what does the function λn. fst (n next2 (pair tru fls)) calculate? Describe the function in a compact description.

    (λn. fst (n next2 (pair tru fls))) m = fst (next2 (… (next2 (next2 (pair tru fls)) …)

    in which next2 is applied m times. As we have seen above, each time we apply next2, we switch tru and fls in the pair. So, after m application, the first element is tru if the m is even and the first element is fls if m is odd.

    So, the function is even(n) or if n%2 = 0 then true else false
7. what is next3 2?

    next3 2 = (λn. (times n n)) 2 = times 2 2 = 4

8. what is next3 4?

   next3 4 = (λn. (times n n)) 4 = times 4 4 = 16

9. what does the function λn. n next3 2 calculate?

   If we call λn. n next3 2 f, we have

   f 1 = 4 = 2^2

   f 2 = 2^2 * 2^2 = 2^4

   f 3 = 2^4 * 2^4 = 2^8

   f 4 = 2^8 * 2^8 = 2^{16}

   In general, f n = 2^{2^n}

Problem 2. Static and Dynamic Scoping. Consider the following program written in C syntax

```
int a , b , c ;    // first declaration

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

int f(int a)       // parameter declaration
{    int b;        // second declaration
     b = a + 1;
     g();          // first call

     { int a;      // third declaration
       int c;      // fourth declaration
       c = b;
       a = b + c;
       g();        // second call
     }

     g();          // third call

     return a + b ;
}

int main()
{
     int a = 2;    // fifth declaration

     a = f(a);
     g();          // fourth call
}
```

1. If static scoping is used, the reference to a in the first call to g() resolves to which declaration?

   first declaration

2. If static scoping is used, the reference to a in the third call to g() resolves to which declaration?

   first declaration

3. If dynamic scoping is used, the reference to a in the first call to g() resolves to which declaration?

   parameter declaration

4. If dynamic scoping is used, the reference to a in the third call to g() resolves to which declaration?

   parameter declaration

5. What is the output of this program if static scoping is used?

In what follows I assume global variables are initialized to 0.

In what follows, I give subscripts to the variable names. a_main refers to the a declared in main, a_f refers to the a declared in f and a_global refers to the global variable a. Other subscripts should be clear from the context. Here is a trace of the execution

```
main()
    a_main = 2
    a_main =        f(a_main) = f(2)  = 5       // see below
                    f(2)
                            a_f = 2
                            b_f = a_f + 1 = 3

                            g()
                                    print a_global b_global c_global            // prints 0 0 0

                            {
                                    c_f_scope = b_f = 3
                                    a_f_scope = b_f + c_f_scope = 3 + 3 = 6
                                    g()
                                            print a_global b_global c_global    // prints 0 0 0
                            }

                            g()
                                    print a_global b_global c_global            // prints 0 0 0

                            return a_f + b_f = 2 + 3 = 5

    a_main =  5

    g()
            print a_global b_global c_global                                    // prints 0 0 0
```

6. What is the output of this program is dynamic scoping is used?

a_global 0[initially]
b_global 0[initially]
c_global 0[initially]

main
a_main 2[1]

1. a_main = 2
2. a_main = f(a_main) = f (2) = 5 // see below
3. f(2)
   a_f 2[argument]
   b_f 3[4]
4. b_f = a_f + 1 = 3
5. g()
6. print a_f b_f c_global // prints 2 3 0

{ a_f_local 6[8]
c_f_local 3[7]

7. c_f_local = b_f = 3
8. a_f_local = b_f_local + c_f_local = 3 + 3 = 6
9. g()
10. print a_f_local b_f c_f_local // prints 6 3 3

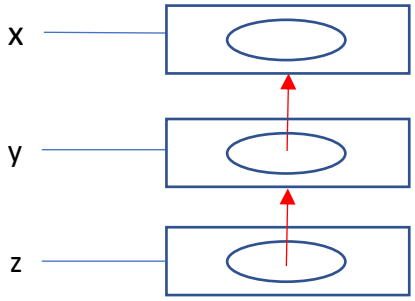
}

11. g()
12. print a_f b_f c_global // prints 2 3 0
13. return a_f + b_f = 2 + 3 = 5
14. g()
15. print a_main b_glonal c_global // prints 5 0 0

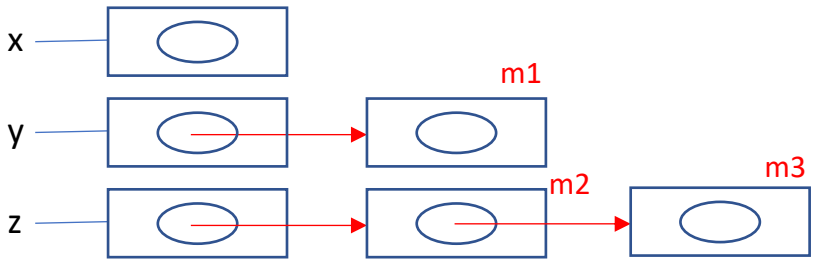
Problem 3. Pointer Semantics in C. Consider the following C code

```
int x;                                  // location 1 associated with x
int *y;                                 // location 2 associated with y
int **z;                                // location 3 associated with z

y = &x;                                 // statement 1
z = &y;                                 // statement 2

y = (int *) malloc(sizeof(int));        // statement 3: m1 allocated
z = (int **) malloc(sizeof(int *));     // statement 4: m2 allocated
*z = (int *) malloc(sizeof(int));       // statement 5: m3 allocated

y = *z;                                 // statement 6
```

1. Draw the box circle diagram after statements 1 and 2 are executed



2. Draw the box circle diagram after statements 1 through 5 are executed



3. Draw the box circle diagram after statements 1 through 6 are executed