**CSE340 FALL SPRING 2021 HOMEWORK 4**

**Due Thursday April 1 (no joking!) 2020 by 11:59 PM**

**PLEASE READ THE FOLLOWING CAREFULLY**

1. Your answers must be typed.
2. On Gradescope, you should submit the answers to separate question separately.
3. Read carefully the required answer format. 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
4. **Note**: The last two problem go over examples related to this homework

**Problem 1 (Lambda Calculus)** The goal of this problem is to give you 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

quad = 𝜆x. 𝜆y. 𝜆z. 𝜆w. pair (pair x y ) (pair z w)

1st = 𝜆q. fst (fst q)

2nd = 𝜆q. snd (fst q)

3rd = 𝜆q. fst (snd q)

4th = 𝜆q. snd (snd q)

tri = 𝜆x. 𝜆y. 𝜆z. pair x (pair y z)

f0 = 𝜆p. pair **(**PLUS (fst p) (snd p)**)** **(**PLUS (scc (fst p))(snd p)**)**

f1 = 𝜆q. quad (2nd q) (3rd q) (4th q) (1st q)

f2 = 𝜆t. tri ( OR (fst t) **(**EQUAL (fst(snd t)) (snd (snd t))**)** ) // first element

( TIMES (fst (snd t)) 2 ) // second element

(snd (snd t) ) // third element

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

1. What is f0 (pair 1 1)?

**Answer.** f0 (pair 1 1) = pair (plus 1 1) (plus 1 2)

= pair 2 3

1. What is f0 (f0 (pair 1 1))?

**Answer** f0 (f0 (pair 1 1) ) = f0 ( pair 2 3 )

= pair (plus 2 3) (plus 3 3)

= pair 5 6

1. What is f0 (f0 (f0 (pair 1 1)))?

**Answer** pair 11 12

1. What does the function 𝜆n. fst (n f0 (pair 1 1)) calculate? Give a *compact* description. Assume n is a Church numeral

**Answer** If we continue we find that f0 (f0 (f0 ( f0 (pair 1 1) ) ) ) = pair 23 24. The question

asks for the first part of the resulting pair after applying f0 n times, starting with pair (1 1). The pattern on the first part of the pair might not be immediately apparent but the second part is 3 \* 2n-1 when n > 0. The first part is (3 \* 2n-1) – 1. So, the answer is:

if n = 0 then 1

else (3 \* 2n-1) – 1

1. what is f1 (quad a b c d)?

**Answer** f1 (quad a b c d) = quad b c d a

1. what is f1 (f1 (quad a b c d))?

**Answer** f1 **(**f1 (quad a b c d)) = f1 (quad b c d a)

= quad c d a b

1. what does the function 𝜆n. 1st (n f1 (quad 0 1 2 3)) calculate? Give a compact description.

**Answer**  The function calculates n mod 4

1. what is tri fls 40 45?

**Answer**  tri fls 40 45 = pair fls (pair 40 45)

1. what is f2 (tri fls 40 45)

**Answer**  f2 (tri fls 40 45) = f2 (pair fls (pair 40 45))

**=** tri (OR fls (EQUAL 40 45)) (TIMES 40 2) (45)

= tri fls 80 45

1. what is f2 (f2 (tri fls 40 45))

**Answer**  f2 (f2 (tri fls 40 45)) = f2 (tri fls 80 45)

= tri fls 160 45

1. what does the function 𝜆n. 𝜆p. 𝜆q. fst (n f2 (Tri fls p q)) calculate? Give a compact description. Assume n is a Church’s numeral greater than 0 and p and q are Church’s numerals.

**Answer** In all applications of the function f2, we get a tri in which the first element is a Boolean and the second and third elements are Church numerals. In each application of f2, the first number (second element of the tri) is multiplied by 2 and the second number is unmodified. If we apply the f2 multiple times, the first number is multiplied repeatedly by 2. After *i* applications, the first number becomes *p*\*2*i* and the second number is unchanged. The Boolean value becomes tru if the two numbers of the tri are equal. Once the Boolean value becomes tru it stays tru in subsequent applications of f2.

Given all of that, the function, which returns the Boolean value of the tri, will return tru if q/p is a power of 2 with an exponent less than or equal to n-1: the function returns tru if q/p = 2*i*, for i < n and returns fls otherwise

**Problem 2 (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

fls indicates that the list is empty, so the second element of the pair should not be accessed

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

tru indicates that the list is not empty, so the second element of

the pair which is ( pair a (pair fls fls) ) contains the list data

a is the first element of the list

(pair fls fls) is the remainder of the list. In this case, is the empty list

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

tru indicates that the list is not empty, so the second element of

the pair which is ( pair a ( pair tru (pair b (pair fls fls) ) ) ) contains the list data

a is the first element of the list

( pair tru (pair b (pair fls fls) ) ) is the remainder of the list. In this case, it is a list that contains one

element which is b

In general, the representation of a list with elements a1, a2, a3, ... , ak in the given order, where **k ≥ 1** is

pair tru (pair a1 L)

where L is the representation of the list whose element in order are a2, a3, ... , ak

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.

**Questions**

1. Write a recursive function that takes a list L as input and produces a list in which every element of L is duplicated. If the input to the function is a list representation of a1, a2, ... , ak, the output should be a list representation of a1, a1, ... , ak-1, ak-1, ak, ak

g = 𝜆dupl. 𝜆L. (isEmpty L)

L

pair tru (pair (fst(snd L) (pair tru (pair (fst(snd L) (dupl (snd(snd L)))

// if the given list has the form a L’ where a is the first element and L’

// is the remainder, the function will output a new list of the form

// a a (duplicate L’)

**Duplicate = fix g**

1. Write a function that takes as input two lists L1 and L2 and returns a new list that is obtain by alternating elements from L1 with elements from L2. If L1 is a list representation of a1, a2, ... , ak , and L2 is a list representation of b1, b2, ... , bk’ , and k > k’, then the result should be a list representation of a1, b1, a2, b2, ... , ak’ , bk’, ak’+1, ak’+2, ... , ak. If k’ ≥ k, the result is a list representation of a1, b1, a2, b2, ... , ak , bk, bk+1, bk+2, ... , bk’.

Again, we write a recursive function.

g1 = 𝜆alter. 𝜆L1. 𝜆L2. (isEmpty L1)

L2

( (isEmpty L2)

L1

pair tru (pair (fst(snd L1) (pair tru (pair (fst(snd L2) (alter (snd(snd L1)) (snd(snd L2)))

)

**Alternate = fix g1**

**Hint** look at the solution of HW3 from Fall 2019 and HW4 from Fall 2020

**Problem 3. Pointer Semantics in C**. Consider the following C program

#include <stdio.h>

#include <stdlib.h>

struct T {

int i;

struct T \* next;

};

struct T \*b[4]; // Global variable. Locations m1 through m4 are

// associated with b[0] through b[3].

struct T \*\*c[4]; // global variable. locations m5 through m8 are

// associated with c[0] through c[3].

int main()

{

b[0] = (struct T \*) malloc(sizeof(struct T)); // location m9 allocated

c[0] = (struct T \*\*) malloc(sizeof(struct T \*)); // location m10 allocated

b[1] = \*c[0];

{ struct T \*a[4]; // a[0] through a[3] are in locations m11 through m14

// point 1

for (int i = 0; i < 3; i++)

{

a[i] = (struct T \*) malloc(sizeof(struct T));// locations m15 through

// m17 allocated in

// successive iterations

b[i+1] = a[i];

c[i] = &a[i+1];

b[i] = \*c[i];

a[i]->next = b[i];

}

// point 2

}

// point 3

free(b[2]);

// point 4

c[0] = &b[0]; // assignment 1

c[1] = &b[1]; // assignment 2

\*c[0] = \*c[1]; // assignment 3

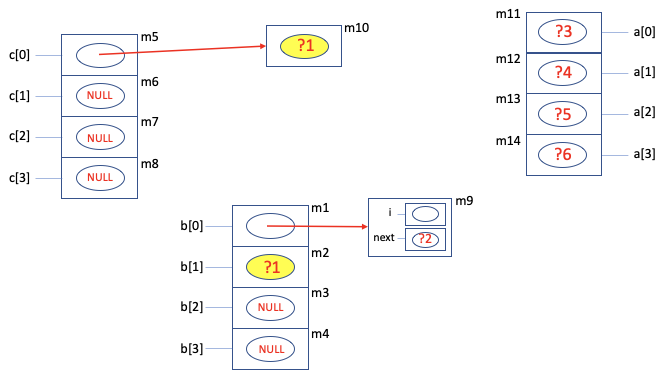
//point 5

}

**Remember**

* A wild pointer is a pointer that is not initialized. It is different from a dangling reference in that a dangling reference points to memory that has been previous allocated and then de-allocated.
* When counting wild pointers, we do not count uninitialized fields in previously deallocated locations or in garbage locations.
* When counting dangling references, we do not count fields in previously deallocated locations or in locations that are garbage.
* Global variables are initialized to 0 in C

**Questions**

Box-circle diagram at Point 1:  


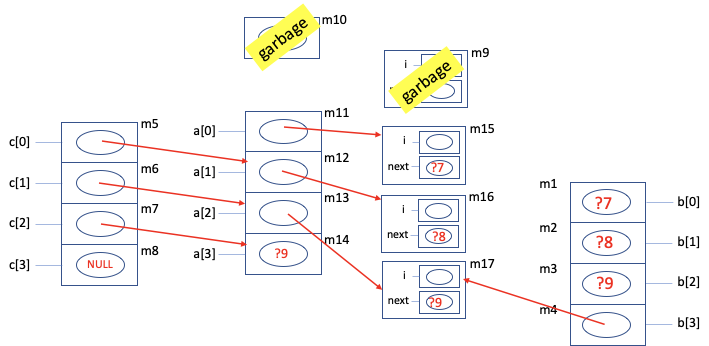
1. What are (if any) the dangling references, the wild pointers and the garbage locations at point 1

No Dangling references

Wild pointers - b[1], \*(c[0]), b[0]->next, a[0], a[1], a[2], a[3]

No Garbage locations

Box-circle diagram at Point 2:



1. What are (if any) the dangling references, the wild pointers and the garbage locations at point 2

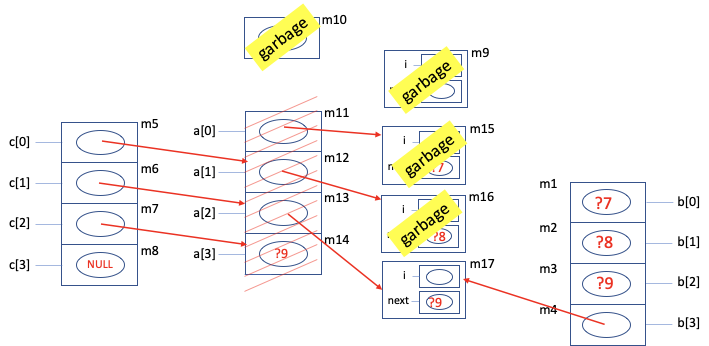
No dangling references

Wild pointers - a[3], b[0], b[1], b[2], a[0]->next, a[1]->next, a[2]->next

Garbage - m9, m10

1. ~~What is an alias of b[0] at point 2? Your answer should be an expression that does not contain “b”. This is a little harder than question 2 above.~~

Box-circle diagram at point 3:



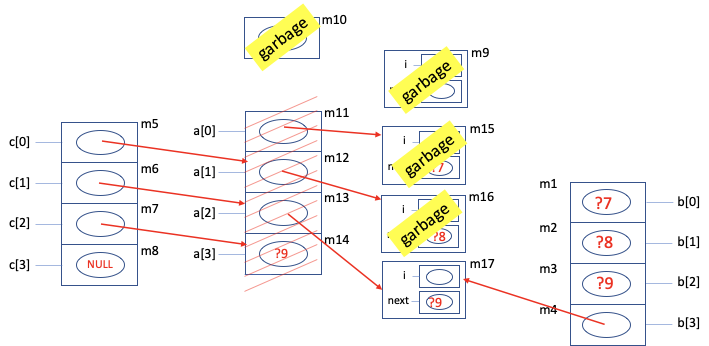
1. What are (if any) the dangling references, the wild pointers and the garbage locations at point 3

Dangling references - c[0], c[1], c[2]

Wild Pointers - b[0], b[1], b[2], b[3]->next

Garbage - m9, m10, m15, m16

Box-circle diagram for Point 4:



1. What are (if any) the dangling references, the wild pointers and the garbage locations at point 4

Dangling references - c[0], c[1], c[2]

Wild Pointers - b[0], b[1], b[2], b[3]->next

Garbage - m9, m10, m15, m16

1. Executing assignment 1 results in an arrow from which location to which location?

m5 to m1

1. Executing assignment 2 results in an arrow from which location to which location?

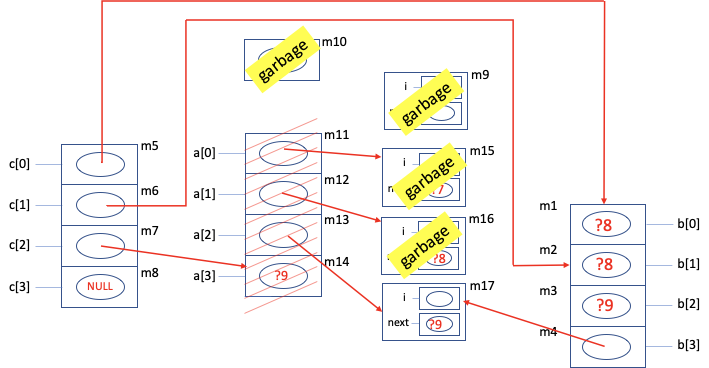
m6 to m2

1. Executing assignment 2 results in an arrow from which location to which location?

m6 to m2

Note: For assignment 3, the arrow will be from m1 to the garbage location pointed to by m2

Box-circle diagram for Point 5:



1. What are (if any) the dangling references, the wild pointers and the garbage locations at point 5

Dangling references - c[2]

Wild Pointers - b[0], b[1], b[2], b[3]->next

Garbage - m9, m10, m15, m16