CSE340 FALL 2020 HOMEWORK 4
Due Monday November 16 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

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)
First = λq. fst (fst q)
Second = λq. snd (fst q)
Third = λq. fst (snd q)
Fourth = λq. snd (snd q)
Tri = λx. λy. λz. pair x (pair y z)
next0 = λp. pair **(**PLUS (fst p) (snd p)**)** **(**PLUS (snd p) (snd p)**)**
next1 = λq. quad (Second q) (Third q) (First q) (Fourth q)
next2 = λt. Tri (OR (fst t) **(**EQUAL (fst (snd t)) (snd (snd t))**)**) // first element
(TIMES (fst (snd t)) 3) // second element
(TIMES (snd (snd t)) 2) // third element

For each of the following, give the value that the expressions evaluates to (note that the question only asks for the answer)

1. What is next0 (pair 1 1)?

Answer

next0 (pair 1 1)
= (λp. pair **(**PLUS (fst p) (snd p)**)** **(**PLUS (snd p) (snd p)**)**) (pair 1 1)
= pair **(**PLUS (fst (pair 1 1)) (snd (pair 1 1))**)** **(**PLUS (snd (pair 1 1)) (snd (pair 1 1))**)**
= pair (PLUS 1 1) (PLUS 1 1)
= **pair 2 2**

2. What is next0 (next0 (pair 1 1))?

Answer

next0 (next0 (pair 1 1))
= next0 (pair 2 2)
= (λp. pair **(**PLUS (fst p) (snd p)**)** **(**PLUS (snd p) (snd p)**)**) (pair 2 2)
= pair **(**PLUS (fst (pair 2 2)) (snd (pair 2 2))**)** **(**PLUS (snd (pair 2 2)) (snd (pair 2 2))**)**
= pair **(**PLUS 2 2**)** **(**PLUS 2 2**)**
= pair 4 4

3. What is next0 (next0 (next0 (pair 1 1)))?

Answer

next0 (next0 (next0 (pair 1 1)))
= next0 (pair 4 4)
= (λp. pair **(**PLUS (fst p) (snd p)**)** **(**PLUS (snd p) (snd p)**)**) (pair 4 4)
= pair **(**PLUS (fst (pair 4 4)) (snd (pair 4 4))**)** **(**PLUS (snd (pair 4 4)) (snd (pair 4 4))**)**
= pair **(**PLUS 4 4**)** **(**PLUS 4 4**)**
= **pair 8 8**

4. What does the function λn. fst (n next0 (pair 1 1)) calculate? Give a *compact* description.

Answer

- next0 (pair 1 1) = pair 2 2
- next0 (next0 (pair 1 1)) = pair 4 4
- next0 (next0 (next0 (pair 1 1))) = pair 8 8
- In general, applying next0 n times starting with (pair 1 1) gives (pair 2^n 2^n)

The function λn. fst (n next0 (pair 1 1)) applies n to next0 and (pair 1 1) and then takes the fst element of the resulting pair. So, we get next0 applied n times to (pair 1 1) = (pair 2^n 2^n) whose fst element 2^n.

The function λn. fst (n next0 (pair 1 1)) calculates $\mathbf{2^n}$.

5. what is next1 (quad fls tru fls tru)?

   Answer

   next1 (quad fls tru fls tru)
   = next1((λx. λy. λz. λw. pair (pair x y) (pair z w))(fls tru fls tru))
   = next1**(**pair (pair fls tru) (pair fls tru)**)**
   = (λq. quad(Second q) (Third q) (First q) (Fourth q)) **(**pair (pair fls tru) (pair fls tru)**)**
   = quad tru fls fls tru

6. what is next1 (next1 (quad fls tru fls tru))?

   Answer

   next1 (next1 (quad fls tru fls tru)) = next1(quad tru fls fls tru)
   = (λq. quad (Second q) (Third q) (First q) (Fourth q)) (quad tru fls fls tru)
   = **quad fls fls tru tru**

7. what does the function λn. First (n next1 (quad fls tru fls tru)) calculate? Give a compact description.

   Answer

   next1 (quad fls tru fls tru) = quad tru fls fls tru
   next1 (next1 (quad fls tru fls tru)) = quad fls fls tru tru
   next1 (next1 (next1 (quad fls tru fls tru))) = quad fls tru fls tru
   next1(next1 (next1 (next1 (quad fls tru fls tru)))) = **quad tru fls fls tru**

   The function λn. First (n next1 (quad fls tru fls tru)) applies n to next1 and **(quad fls tru fls tru)** and returns the First of the resulting quad. From above, we see that the First of the resulting quad is true for 1, 4, 7, 10, 13, … . In general, the function returns **tru** if (n-1) is a multiple of 3 and returns **fls** otherwise. In other words, the function calculates (n -1) % 3 == 0.

8. what is Tri fls 40 45?

   Answer

   Tri fls 40 45 = (λx. λy. λz. pair x (pair y z)) fls 40 45

   = p**air fls (pair 40 45)**

9. what is next2 (Tri fls 40 45)

   Answer

```
next2 = λt.  Tri ( OR (fst t) (EQUAL (fst (snd t)) (snd (snd t))) )      // first element
                 (TIMES (fst (snd t)) 3)                                  // second element
                 (TIMES (snd (snd t)) 2)                                  // third element
```

next2 (Tri fls 40 45)

= next2(pair fls (pair 40 45))
= Tri**(**OR (fls) (EQUAL 40 45)**)(**TIMES 40 3**)(**TIMES 45 2**)**
= Tri (OR fls fls) (120) (90)
= **Tri fls 120 90**
= Tri fls 40*3 45*2

10. what is next2 (next2 (Tri fls 40 45))

Answer

next2 (next2 (Tri fls 40 45))
= next2 (Tri fls 120 90)
= Tri**(**OR(fls)(EQUAL 120 90)**)(**TIMES 120 3**)(**TIMES 90 2**)**
= Tri fls 360 180
= Tri fls 40*3^2 45*2^2

11. what does the function λn. λp. λq. fst (n next2 (Tri fls p q)) calculate? Give a compact description.

Answer

The function λn. λp. λq. fst (n next2 (Tri fls p q)) applies n to next2 and **(Tri fls p q)** and returns the fst of the resulting Tri.

We first note that once the first element of the Tri becomes tru, it stays tru in subsequent applications of next1 to it.

In general, applying the function λn. λp. λq. fst (n next2 (Tri fls p q)) i times to **(Tri fls p q)** gives (Tri b $p*3^i$ $q*2^i$), where b is a boolean value. If for some i, $p*3^i = q*2^i$, then for i+1, b will be **tru**. If $p*3^i = q*2^i$, then $p/q = 2^i/3^i = (2/3)^i$. In order for b to be tru after n applications of next2, i must be less than n. So, the function calculates if $p/q = (2/3)^i$ for some $i < n$.

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 $a_1, a_2, a_3, \ldots, a_k$ in the given order, where $\mathbf{k \geq 1}$ 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.

Questions

1. Write a recursive function that shifts a list right. If the input to the function is a list representation of $a_1, a_2, \ldots, a_k$, the output should be a list representation of $a_k, a_1, \ldots, a_{k-1}$
2. write a function that takes as input a list L and a Church numeral n and returns a list which is obtained from L by shifting L to the right n times. You can assume that you have a function that shifts write (part 1) in answering this part.

Hint look at the solution of HW3 from Fall 2019.

1. I start with a high level overview of the answer in pseudocode and then I give the lambda calculus expressions.

   If we have a list L that we want to *shiftRight*, there are three cases to consider

   a. the list is empty: in this case *shiftRight* **L** = **L**
   b. the list has only one element: in this case *shiftRight* **L** = **L**
   c. the list has more than one element: in this case the list has the form **a L'** where a is the first element of **L** and **L'** is a list that contains at least one element. To *shiftRight* **L**, we need to drop the last element of **L'** to obtain **L''**, insert **a** at the end of **L''** to obtain **L'''** and make a list that has the last element of **L** as its first element and **L'''** as its second element.

   To implement the solution in lambda calculus, we introduce some helper functions (some of them are from HW3 Fall 2019).

```
//-----------------------------------------------------------------------------------------------------------
// Empty List is a function that returns a boolean value which is tru if the argument is an empty
// list and fls if the argument is not an empty list
//-----------------------------------------------------------------------------------------------------------
```
EmptyList = λL. (fst L) fls tru

```
//-----------------------------------------------------------------------------------------------------------
// FirstElement is a function that only works for a non-empty list. It returns the first element of the
// list
//-----------------------------------------------------------------------------------------------------------
```
FirstElement = λL. (fst (snd L))

```
//-----------------------------------------------------------------------------------------------------------
// AddElement is a function that takes a list L and an element a as arguments and returns a new
// list in which the first element is a and the tail is L
//-----------------------------------------------------------------------------------------------------------
```
AddElement= λa. λL. pair tru (pair a L)

```
//-----------------------------------------------------------------------------------------------------------
//LastElement (recursive) is a function that returns the last element of a list. The function expects
// a non-empty list as argument. This function is recursive, so we define it by first defining a
// function g with argument LastE and then we set LastElement = fix g
// The function works as follows. If L has only one element ( (snd(snd L) is empty), it returms
// the first element of L, otherwise, it returns the last element of (snd (snd L)).
//-----------------------------------------------------------------------------------------------------------
g = λLastE. λL. ( EmptyList (snd(snd L)) )
                        ( FirstElement L )
                        ( LastE (snd (snd L)) )
```

LastElement = fix g

```
//-----------------------------------------------------------------------------------------------------------------
//LastElement (recursive) is a function that returns the last element of a list. The function expects
// a non-empty list as argument. This function is recursive, so we define it by first defining a
// function h with argument dropLastElem and then we set LastElement = fix g
//-----------------------------------------------------------------------------------------------------------------
h = λdropLastElem. λL. (EmptyList L)
                              (pair fls fls)
                                      ((EmptyList (snd(snd L)))
                                        (pair fls fls)
                                       (pair tru (pair (fst(snd L)) (dropLastElement snd(snd L)))
                         )
```

DropLastElement = fix h

```
//-----------------------------------------------------------------------------------------------------------------
// shiftright is the function we are asked to write. The following implenebts the high-level
// description that I gave above
//-----------------------------------------------------------------------------------------------------------------
shiftRight = λL. (EmptyList L)
                                    (pair fls fls)
                                    ( (EmptyList (snd ( snd L)))
                                                 L
                                                 (AddElement (LastElement L)) (DropLastElement L))
                                    )
```

2. To shift right n times, we apply n to the shiftRight function. We get: λL. λn. (n shiftRight L)

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

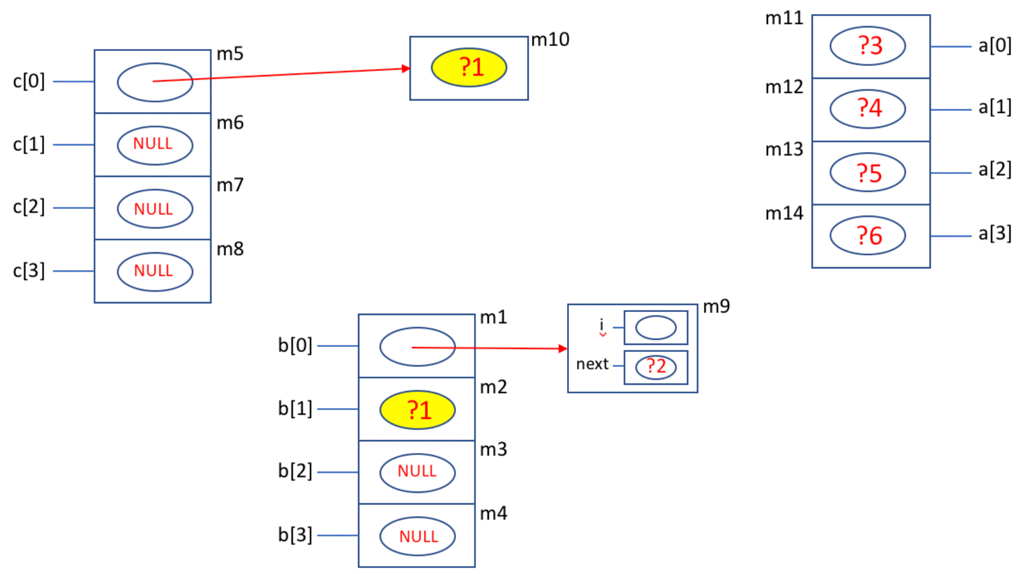
```
#include <stdio.h>
#include <stdlib.h>

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] = &b[i+1];
           b[i]->next = *c[i];
     }
     // point 2
   }
   // point 3

   free(*c[2]);
   // point 4

   b[2] = b[1];    // assignment 1
   *c[2] = *c[0];  // assignment 2
   //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:

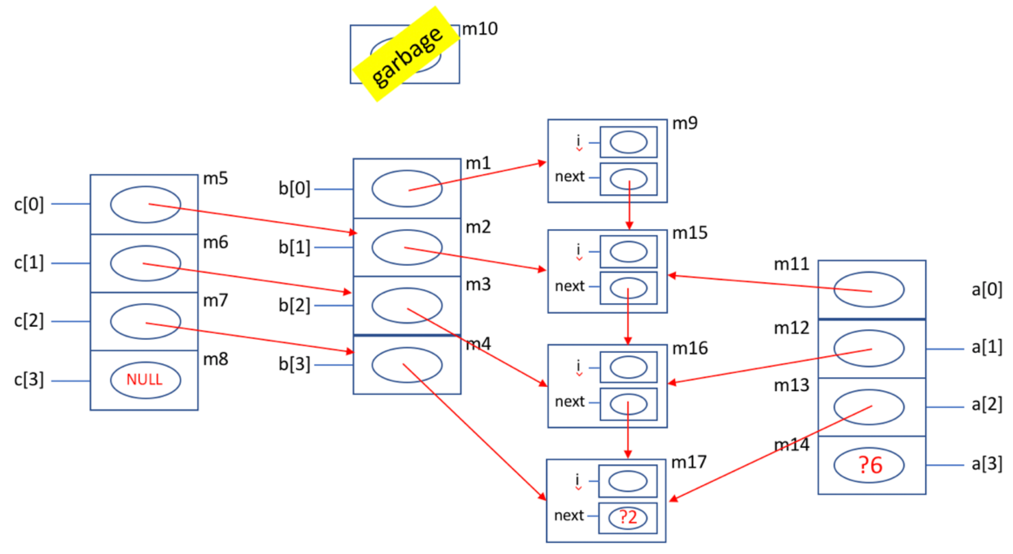


Note that global variables are initialized to 0 in C. Local variables (like a[]) are not guaranteed to be initialized and they should be treated as not initialized.

1. What are (if any) the dangling references, the wild pointers and the garbage locations at point 1
   Answer
   - No dangling references
   - Wild Pointers: b[1], *(c[0]) , b[0]-> next, a[0], a[1], a[2], a[3]
   - No garbage locations

2. What is an alias of b[1] at point 1? Your answer should be an expression that does not contain "b".
   Answer
   Question was cancelled

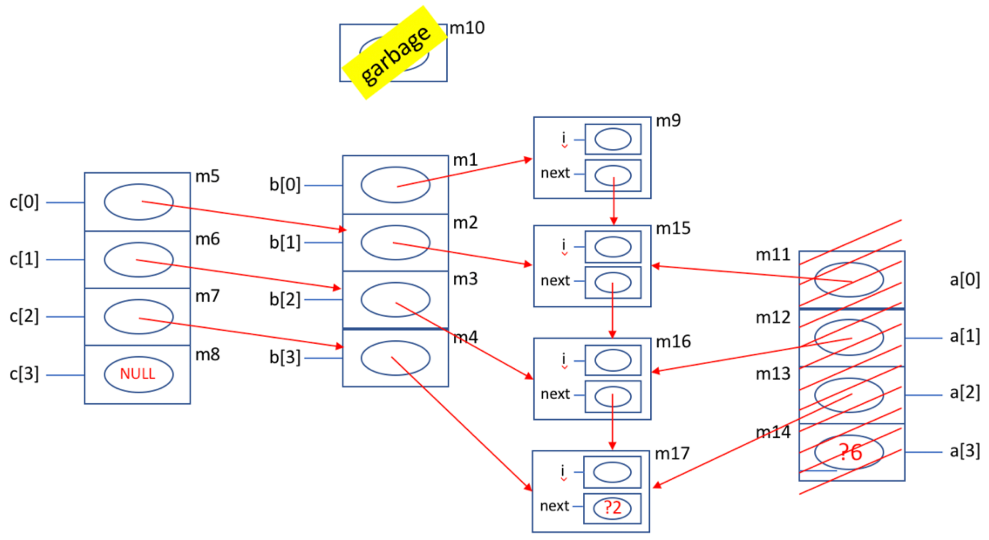
   Box Circle Diagram at Point 2:



3. What are (if any) the dangling references, the wild pointers and the garbage locations at point 2
Answer
   - No dangling references
   - Wild pointers: a[3], b3->next, (or a[2]->next)
   - Garbage locations: m10

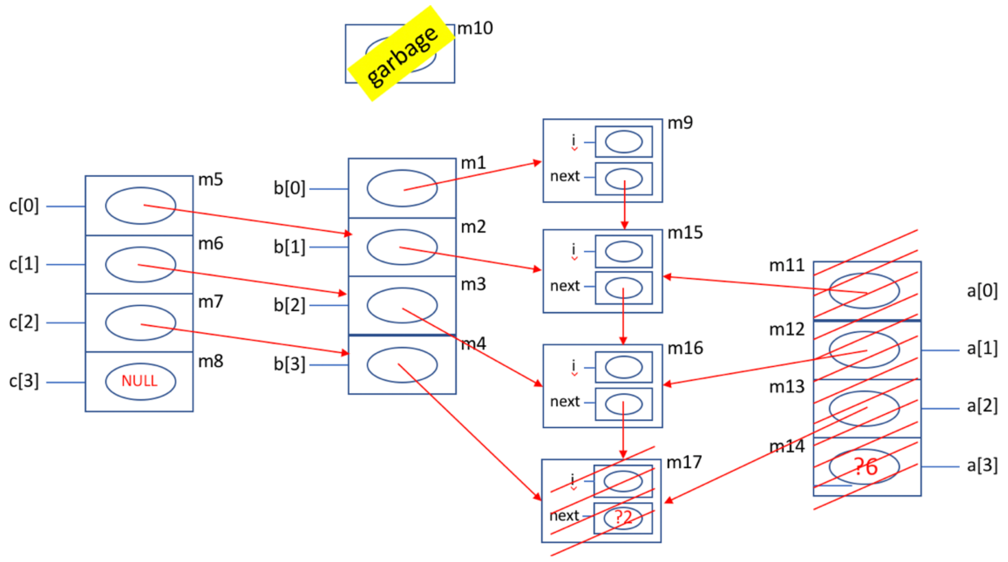
4. What is an alias of b[0] at point 2? Your answer should be an expression that does not contain "b". This is a little harder than question 2 above.
*(c[0] - 1)

Box Circle Diagram at Point 3:



5. What are (if any) the dangling references, the wild pointers and the garbage locations at point 3
   - No dangling references
   - Wild pointer: b[3]->next
   - Garbage locations: m10

Box Circle Diagram at Point 4:

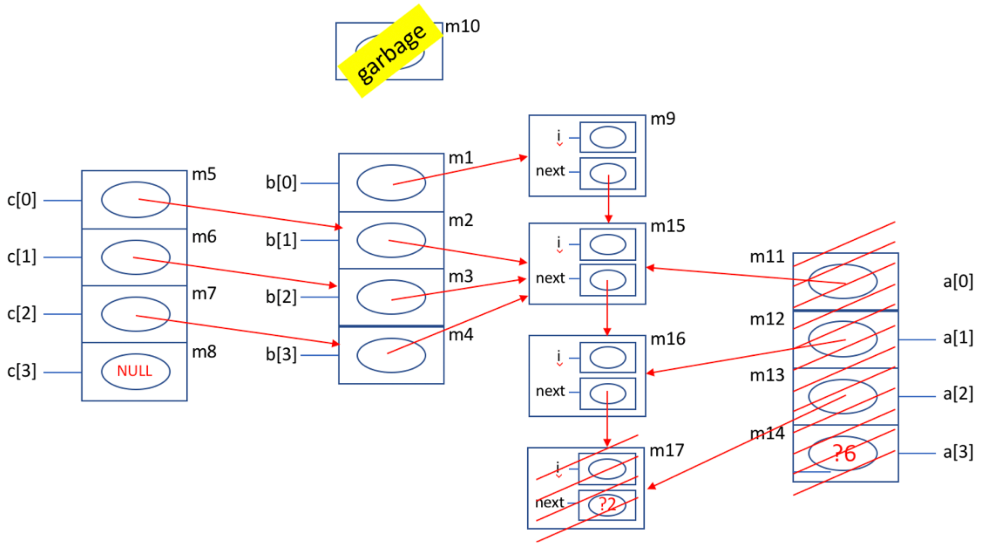


6. What are (if any) the dangling references, the wild pointers and the garbage locations at point 4
   - Dangling reference: b[3], (*b[2]).next (or b[2]->next)
   - Wild pointers: none
   - Garbage: m10

7. Executing assignment 1 results in an arrow from which location to which location?
   m3 to m15

8. Executing assignment 2 results in an arrow from which location to which location?
   m4 to m15

   Box Circle Diagram at Point 5:



9. What are (if any) the dangling references, the wild pointers and the garbage locations at point 5
   - Dangling reference: (*((*b[3]).next)).next
     (or b[3]->next->next, or b[2]->next->next or b[1]->next->next)
   - Wild pointers: none
   - Garbage: m10